NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

**WEB-BASED DATABASE APPLICATIONS:
AN EDUCATIONAL, ADMINISTRATIVE
MANAGEMENT SYSTEM FOR MILITARY ACADEMIES**

by

Rasim Topuz

March 2002

Thesis Co-Advisors: Thomas Otani
William Haga

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Web-based Database Applications: An Educational, Administrative Management System for Military Academies			**5. FUNDING NUMBERS**
6. AUTHOR: Topuz, Rasim			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT ***(maximum 200 words)***

Not only does a military academy have all the information overload of a normal university but it also has the extra burden of the military environment. Without a reliable information system, administrative and educational functions cannot be performed.

This thesis deals with the problem of administrative overload in managing student, faculty, regiment personnel and course data in a military academy. It proposes an Educational Administrative Management System (EAMS), a Web-based data management system, as a solution. With this goal in mind, existing client-server architectures, server side application development tools and database technologies are explored, and the best configuration of these tools is selected. Some of them are recommended.

As a result of the study, Java Servlets and Java Server Pages are found as the optimal server-side programming tool for the application. A working prototype of the system is provided based on Oracle 8i DBMS, Apache Tomcat Web server, Java Servlets and Java Server Pages.

Suggestions are provided for coping with change management issues during the implementation of the system.

14. SUBJECT TERMS Internet, Web Based Architecture, Java, Java Servlets, Java Server Pages (JSPs), Structured Query Language (SQL), Java Database Connectivity (JDBC), Database and Military Academy			**15. NUMBER OF PAGES** 267
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

WEB-BASED DATABASE APPLICATIONS: AN EDUCATIONAL, ADMINISTRATIVE MANAGEMENT SYSTEM FOR MILITARY ACADEMIES

Rasim Topuz
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN COMPUTER SCIENCE
AND
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
March 2002

Author: Rasim Topuz

Approved by: Thomas Otani
Co-Advisor

William Haga
Co-Advisor

Chris Eagle
Chairman, Department of Computer Science

Dan Boger
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Not only does a military academy have all the information overload of a normal university but it also has the extra burden of the military environment. Without a reliable information system, administrative and educational functions cannot be performed.

This thesis deals with the problem of administrative overload in managing student, faculty, regiment personnel and course data in a military academy. It proposes an Educational Administrative Management System (EAMS), a Web-based data management system, as a solution. With this goal in mind, existing client-server architectures, server side application development tools and database technologies are explored, and the best configuration of these tools is selected. Some of them are recommended.

As a result of the study, Java Servlets and Java Server Pages are found as the optimal server-side programming tool for the application. A working prototype of the system is provided based on Oracle 8i DBMS, Apache Tomcat Web server, Java Servlets and Java Server Pages.

Suggestions are provided for coping with change management issues during the implementation of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

This thesis, a result of my two years of graduate studies at the Naval Post Graduate School, is dedicated to my parents who encouraged me at every step of my life, to my brothers Rasih, Kazim and Ender who had more confidence in me than I had in myself, to my beautiful new baby daughter, Ilayda Nur—and most importantly to my dear wife, Fazilet, who unflaggingly supported and tolerated me while I prepared the research project.

I would also like to express my sincerest thanks to all the people who have helped me along the way in preparation of this thesis. Among them I would like to thank Ron Russell for his editorial help. Finally, I would particularly like to thank my outstanding advisors; Professor Thomas Otani and William Haga, for their patience, assistance and guidance in helping me prepare this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

1. Problem

Not only does a military academy have all the information overload of a normal university, but it also has the extra burden of the military environment. Without a reliable information system, administrative and educational functions cannot be performed.

Currently most academic organizations use an information system based on a database to handle this problem. But in most cases, because of the system architecture, only the administrative office personnel has access to the information, which wastes computer resources, manpower and time, and generates additional paperwork.

2. Solution Proposed by This Thesis

Even though all the information is in the system, as only the administrative office personnel has access to this system, any other users who need the information do not have access to the system. A solution is an Educational Administrative Management System (EAMS), which will provide access to all information on a need-to-know basis. All the users in the military academy, such as the students, faculty members, department heads, the command personnel as well as the administrative office personnel, will have access to the system, based on their privileges.

3. Consequence If the Problem Is Not Solved

Without implementing this system, duplicate files are kept in different locations in the organizations. As a result, one must update too many files or data for the same information, wasting much manpower. Moreover the safety of the information is jeopardized, currently a major concern of all organizations.

4. Specific Case

The Turkish Naval Academy is currently using an online Student Administrative Information System. This system is only accessible by the administrative office personnel. The officers in the regiment or the faculty members in the academic departments do not have direct access to the system and the information. The author believes that this kind of information infrastructure results in redundant and imprecise

data maintained at different field sites by different personnel. For example, if the department heads want to access the current grades of the students, they need to request this information from the administrative office, and a written report will be submitted in response. This requires much time and extra paperwork.

Another specific example is the regiment personnel who command a student unit in the academy. If an officer needs some particular information (such as the discipline points or a current address) the officer must request this information from the administrative offices in the regiment building. Once again the response will be submitted on paper. To reduce this extra time, most of the officers also keep extra files in their offices, which creates extra paperwork and requires extra time. Also as the administrative offices are only open during business hours, special extra files must be kept and updated in special offices or on computers for the faculty members working on a project late in the academy or any regiment officer on duty who wants to access information about the personnel. This wastes computer resources, manpower and time.

The proposed solution to the academy is a new Educational Administrative Management System, which will enable the faculty members, command personnel, students, department heads and the administrative office personnel access to the system. In this system, the officers in the regiment, even the regiment or academy commander, can locate information about any student or person with a few mouse clicks. For the faculty members or the department heads seeing the level of the students is easy. Their grades, course and information about their instructors will be just a few mouse clicks away.

As the system will be implemented as a Web-based system, which the users will access via their browsers, the system will also have an advantage of being portable. The system will be accessed anywhere and anytime, which is believed to enhance the productivity of the personnel. Even with the necessary security precautions, the users can access the system with their PDAs. The proposed system is aimed to be an intermediary between the user and the information. With the proposed system, most of the current problems will be eliminated, resulting in considerable savings of personnel power and time while proving the information to the users quickly.

B. OBJECTIVES

The goal of this thesis is to design, implement and analyze a Web-based Educational Administrative Management System (EAMS) for the Turkish Naval Academy. In keeping with this goal, the current architecture types, development tools and database technologies will be discussed and the best configuration of these tools will be selected. According to this selection, a prototype for the naval academy will be designed and implemented.

The primary research objective of the thesis is to provide an object-oriented, Web-based, data-retrieval system for the naval academy that will enable access by the various users. In the research, the following questions will be addressed:

- What are the initial requirements for the EAMS application for a military academy?
- What are the existing information system architecture types, and what are the benefits of the three-tier client server architecture?
- What is the appropriate tool for developing a Web-based application?
- What are the possible impacts of implementing EAMS in the academy from the view of the change management?

C. LIMITATIONS

A Web-based EAMS application prototype will be implemented; however, to implement the prototype rapidly, the initial requirements of the system will not be defined in detail. According to the current official reports used in the academy, more detailed interfaces could be needed to generate system outputs directly as official reports. From the system manager perspective, the interfaces for auditing the system online, checking the user's activities and other issues related with security will not be implemented in the initial prototype. From the database migration perspective, the migration problems, related to the original database and the EAMS, will not be covered in this thesis.

D. METHODOLOGY

The thesis research will cover the following phases:

- **Requirements Analysis**: The analysis of the current system and the initial requirements will be defined in this phase.
- **Architectural Design:** The architecture of the system will be selected and designed in this phase.
- **Application Development Tools Evaluation:** In this phase, the best development tool will be selected from the current development technologies.
- **Logical and Physical Database Design:** The selection of the entities for the system and the related tables/relations will be defined and designed in this phase. Also for the implementation purposes, a database instance will be created and populated with data.
- **Application Design and Implementation:** In this phase, the Web-based three-tier Educational Administrative Management System (EAMS) prototype for the Naval Academy will be designed and built. The configuration of the system and the possible impacts of the implementation, such as the resistance to change, will also be studied in this phase.
- **Analysis:** In this phase, the benefits of the system will be identified and the screen snapshots of the prototype will be explained. Possible future enhancements will also be discussed.

E. ORGANIZATION OF THE THESIS

This thesis consists of four general parts. Each part is based on the previous one: Part I addresses the problem, analyzes the current situation, and lists the requirement for the proposed system. Part II gives the reader the necessary technological background information required to understand the other parts of the research. This part provides an overview of three-tier and multi-tier applications, server side programming technologies,

relational databases and Structured Query Language (SQL), Java and Java Database Connectivity (JDBC). Part III describes the design and implementation of the EAMS prototype. Part IV provides a conclusion to the reader.

The following is a more detailed outline for each chapter:

Chapter I. Introduction: This chapter introduces the problem to be addressed, gives a justification and purpose for this work and lists the basis structure of the thesis.

Chapter II. Planning and Design: This chapter describes the mission and vision of the research, the current system, its strengths and weaknesses and the requirements of the proposed system.

Chapter III. Architecture Evaluation: This chapter contains an overview of two-tier and three-tier architectures.

Chapter IV. Evaluation of Server Side Technologies: In this chapter, the current server-side programming technologies and their advantages/drawbacks are discussed.

Chapter V. Relational Databases and the Structured Query Language: This chapter gives an overview on Relational Databases and the Structured Query Language (SQL).

Chapter VI. SQL Application Programming Interfaces and Java Database Connectivity: This chapter first gives a general overview on SQL Application Programming Interfaces and later gives detailed information on Java Database Connectivity (JDBC).

Chapter VII. Design and Implementation of the EAMS: In this chapter, first a data model is built, and later it is transformed into a relational database. Then by going through the application design process, the prototype for the EAMS will be designed and implemented. Later the screen snapshots of the prototype are listed and commented on. Finally the impact of the implementation, such as the resistance to change, is identified in this chapter.

Chapter VIII. Conclusions: This chapter summarizes the lessons learned, and gives directions on how to enhance the application prototype and on how to guide the reader for future related work.

II. PLANNING AND DESIGN

This chapter details the overall planning process used in the development and fielding of the application. This planning process incorporates the project's mission, vision, values and goals. The chapter first summarizes the current application and its strengths and weaknesses. Later, the requirements of the system are defined as the first step in the design of the application.

A. HIGH LEVEL PERSPECTIVE

1. Plan

Information technology offers a variety of services for all stakeholders of a military academy. Implementing these services and applications will help increase the value of the system, reduce the paperwork and enhance service. An "Educational Administrative Management System" (EAMS) specifically designed for the military academy, can meet all the stakeholders' requirements.

a. Mission

Research, analyze, design, implement and document a Web-based "Educational Administrative Management System" based on the defined requirements. This application will provide current, accurate information, which will be available anywhere on a 24-hour basis in the following areas:

- For Faculty Personnel: to maintain, update and access the information about all courses and all students.
- For Department Heads: to maintain, update and access information on the departments' administrative tasks.
- For Regiment Personnel: to maintain, update and access information on the military and administrative tasks and the command of the regiment units.
- For students: to access information so they can fulfill their academic duties and military training.

b. Vision

The vision of the EAMS is to provide access to information about the specific duties to reduce paperwork, wasted time and to improve the decision-making process for the Naval Academy by enhancing the communication and allowing access to the data anytime and anywhere.

c. Values

The application will conform the highest standards of honesty, integrity, accuracy and professionalism. Information will be provided to the users in a manner that reflects the professional ethic of the Naval Academy with the highest reliability and security.

d. Goals

The goals of the EAMS can be summarized as follows:

- Conveniently and consistently support the Naval Academy's informational, educational, and administrative tasks and their requirements.
- Providing the necessary information to the users on a need-to-know basis anytime and anywhere.
- Giving value to the students by including them in the system as users, who will access the relevant information about their academic and military achievements and current situations.

2. Current Application

a. History and Description

The Naval Academy started to use a LAN based, online Information Management system during the early 1980s. This system was a combination of small workstations connected to the mainframe computers. That system was so large that it covered the whole campus. During the mid-1990s, as the information requirements of the academy were changing, the system could not meet the academic and administrative needs.

For this reason in the late 1990s, the simple workstations were changed to Windows NT and Sun workstations, which were connected to the databases and servers. The workstations are stationed mostly in the registrar and administrative offices where the secretaries or administrative office personnel access and update the information and later the hardcopies are given to the related personnel per request.

b. Strengths and Weaknesses of the Current System

Since the information system was started during the mid-1980s, the Naval academy has an IT background on the educational and administrative system. The inner and the outer security of the information center were defined in details and maintained strictly. The accuracy and the consistency of the information were planned in a perfect manner so that no problems occurred from the beginning of the systems.

The current system is accessed by a small number of users in specified locations. This ensures the security of the system. Even though all the data is available, only the administrative office personnel have access to the information online. Even though the data exists on the system, the printouts of the information are provided to other personnel. The need-to-know principle was achieved by minimum user access to the system, and minimum prints to the related personnel.

For the safety of the sensitive information, such as students' course grades, the faculty personnel have access to special workstations to enter the grades in the system. But this access is merely for entering the data. They do not possess a special account allowing them to receive personal information on the courses or students, such as the students' photographs. The regiment personnel or the students do not have direct access to the system. To provide the information to the other personnel, such as regiment command personal or administrative faculty personnel, classic files and folders are also kept in the administrative offices, which creates an extra burden for the office personnel.

B. INITIAL DIRECTION

1. The Application

The academy is currently using Windows NT and Sun workstations. Because of this platform diversity in the academy, one of the main requirements of the application is to be implemented with a portable language, which must be platform independent.

The new application will provide the following personnel access the system:

- Faculty Personnel
- Department Heads
- Students
- Regiment Personnel
- Administrative Office Personnel

The users will have specific user ID's and passwords to access this system. Users will have specific privileges and according to these privileges their access and their system tools in the system will be different.

2. The Database

Even though the academy has a reliable database and no problems have occurred, in keeping with the project, the database technologies will be discussed and a sample database will be built and populated for the implementation purposes. The application will introduce new technologies to the academy but these technologies will work with the existing database. A new database is not necessarily needed.

C. REQUIREMENTS

1. General Application Architecture

- The application should be designed on a Web-based architecture so that reliable access from anywhere on the campus can be achieved.
- The language should be platform independent and must provide easy implementation of the Software Engineering concepts and principles,

providing the software reuse, object-oriented structure, maintainability and reliability.

- A prototype of the application should be provided. The later changes in the requirements and user interfaces will be made based on this prototype.

2. User Profiles

The application should provide access to the following users:

- Faculty Members
- Department Heads
- Regiment Personnel
- Students
- Administrative Office Personnel

3. Use Cases

The application should perform the following use cases and functions.

a. Faculty

- The faculty user will access the system by entering their login ID and password on the login page.
- A menu of available tools for the faculty user will be shown on the screen.
- By selecting specific menu options, the users
    - Will see the courses that they are giving and the sections that are offered.
    - Will see personal information about themselves and will have the ability to update or change the data.
    - Will see the students taking their courses and their grades for the specific courses.

- Will have the ability to update the grades of the students.
- Will see the class schedule for that semester.
- Will see the specific announcements that are addressed to the faculty members.

b. Department Heads

- The Department Heads will access the system by entering their login ID and password on the login page.
- A menu of available tools for the Department Heads will be shown on the screen.
- By selecting specific menu options, the users
  - Will see the courses offered in their department.
  - Will see the instructors in their department.
  - Will see the students who are taking the courses in their department.
  - Will see the grades of the students who are taking the courses from their department.
  - Will see their own personal information and will have the ability to update or change the data.
  - Will see the specific announcements that are addressed to the department heads.

c. Regiment Personnel

- The Regiment personnel users will access the system by entering their login ID and password on the login page.
- A menu of available tools for the Regiment personnel users will be shown on the screen.
- By selecting specific menu options, the users

- Will see the units they are commanding.
- Will see the officers' information in their unit(s).
- Will see the students' personal, educational (courses and grades) and administrative (punishments, awards and the disciplinary points) information in their unit(s).
- Will see their own personal information and have the ability to update or change the data.
- Will see the specific announcements that are addressed to the regiment personnel.

d. Students

- The Students will access the system by entering their login ID and password on the login page.
- A menu of available tools for the Students will be shown on the screen.
- By selecting specific menu options, the users
  - Will see the courses, they are taking and their corresponding faculty members.
  - Will see their grades for a specific semester.
  - Will see their company information, such as their commanders, punishments, awards and current disciplinary points.
  - Will see their own personal information and have the ability to update or change the data.
  - Will see the specific announcements that are addressed to the students.

e. Administrative Office Personnel

- These users will access the system by entering their login ID and password on the login page.

- A menu of available tools will be shown on the screen. These menu options will be designed according to their office unit.
- By selecting specific menu options, the user will access the necessary data and have the ability to update or change the information.
- The administrative office personnel are responsible for official reports. In order to design these interfaces and system outputs, the requirements must be defined in more detail, which can only be done in the academy by looking at the format of the original official reports. For this reason, the interfaces of this user type will not be implemented in the prototype.

III. ARCHITECTURE EVALUATION

This chapter defines the basic concepts of client/server architecture, describes the two-tier and three-tier architectures and analyzes their perspective benefits and limitations.

A. DEFINITION

The client/server model "entails two autonomous processes working together over a network; the client processes request specific services which the server processes respond to and process" (Orfali, 1996).

The majority of end-user applications consist of three components: presentation, processing, and data. The client/server architectures can also be defined by how these components are split up among software entities and distributed on a network. Whitten (2001) defines the client/server architecture as a solution in which the presentation, presentation logic, application logic, data manipulation, and data layers are distributed between client PCs and one or more servers.

The term client/server dates back to the 1980s and refers to personnel computers on a network. This architecture is intended to improve usability, interoperability and scalability of enterprise information systems.

B. TWO-TIER ARCHITECTURE

Although a two-tier client/server can be designed in several ways, we will focus on examining what is overwhelmingly the most common implementation. (Five styles of client-server architectures can be found in Appendix A.) In this implementation, the three components of an application (presentation, processing, and data) are divided between two entities: the client and the database server (Figure 1).

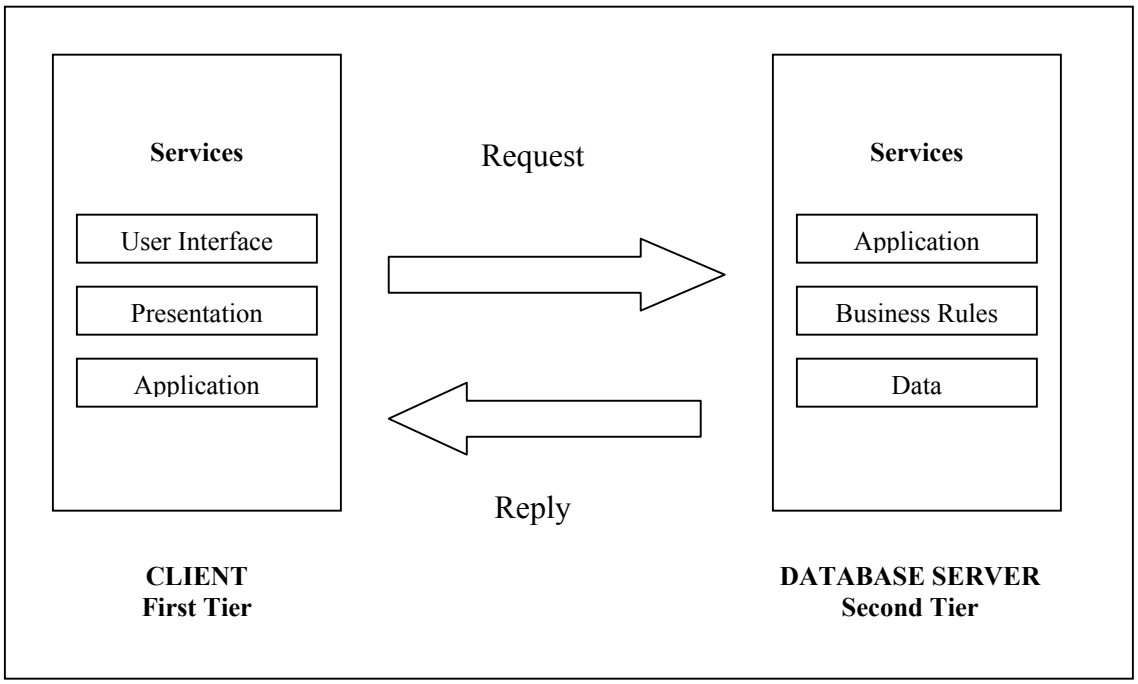

Figure 1. Two-Tier Architecture

In this architecture, the presentation is handled exclusively by the client, processing is split between client and server, and data are stored and accessed via the server. The PC client assumes the bulk of the responsibility for the application (functionality) logic with respect to the processing component while the database engine handles the data intensive tasks.

In such a data access topology, a data engine would process requests sent from the clients. Sending data process requests from client to server requires a tight linkage between the two layers. The client must know the syntax of the server, the location of the server, and how the data are organized and named.

One of the advantages of a two-tier environment is the application development speed. In most cases, a two-tier system can be developed in a small fraction of time that it would take to code a comparable but less flexible legacy system. (Gallaugher, 1996)

Since the bulk of the application logic exists on the PC client, the two-tier architecture faces a number of potential version-control and application-redistribution problems. A change in business rules would require a change to the client logic in each application portfolio, which is affected, by the change. Modified clients would have to be

redistributed through the network. This creates an extra burden for administrators and makes the update prone to error. The clients that fail to be updated will get incorrect information or in some cases may never get access to information on the server.

C. THREE-TIER ARCHITECTURE

The three-tier architecture attempts to overcome some of the limitations of the two-tier scheme by separating the business application into three logical components: presentation, application logic, and data management. These logical components are "clean layered" in such a manner that each runs on a different machine or platform and communicates with the other components via the network. (Goodyear, 2000)

In this client/server model, all the presentation logic resides on the client, all the application logic resides on multiple back-end application servers, and all the data management logic resides on multiple back-end database servers. (Figure 2)

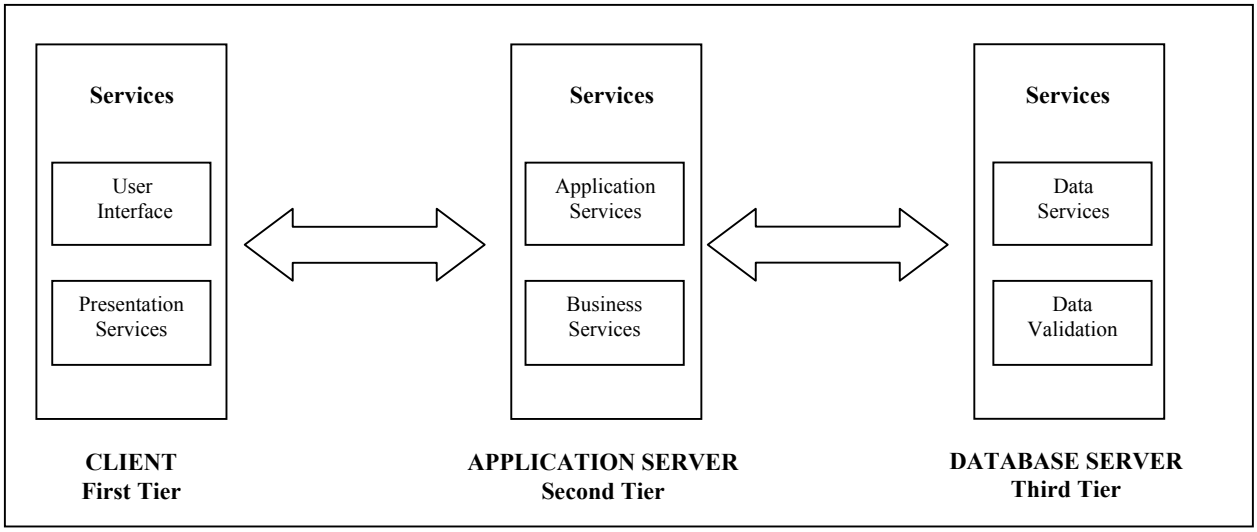

Figure 2. Three-Tier Architecture

When a specific process or data access is required by the presentation client, a call is made to a middle-tier application server. This tier can perform the process or perform requests to the other servers as a client.

In addition to this openness stated above, several other advantages are presented by this architecture. Having separate software entities can allow for the parallel development of individual tiers by the application specialists. Having experts focus on each of these layers can increase the overall quality of the final application.

Modularly designed middle-tier code components can be reused by several applications. Reusable logic can reduce subsequent development efforts, can minimize the maintenance workload, and can decrease migration costs when switching client applications.

The three-tier architecture also provides for more flexible resource allocation (Edwards, 1999). Middle-tier application servers are portable and can be dynamically allocated and shifted as the needs of the organization change. Multiple server requests and complex data access can emanate from the middle tier instead of the client, further decreasing traffic. Also, since PC clients are now dedicated to just presentation, memory and disk storage requirements for PCs will potentially be reduced.

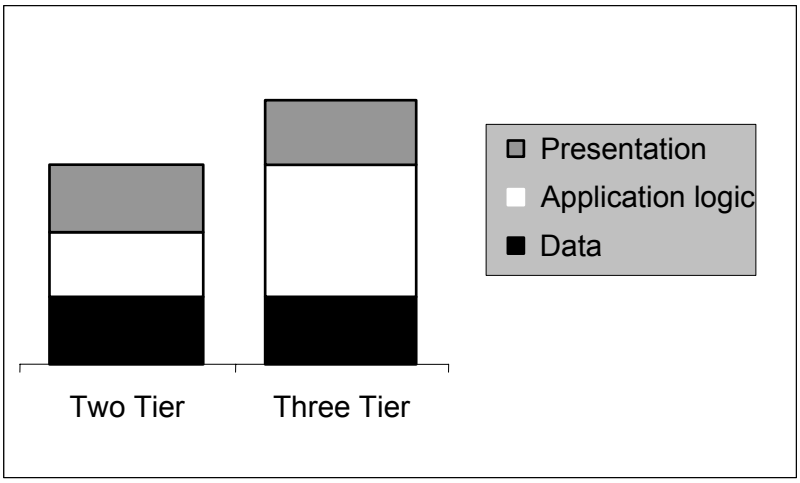

Figure 3. Comparison of Initial Development Effort (From Gallaugher, 1996)

From the initial development effort perspective, the three-tier application takes longer to develop when compared with the two-tier application. (Figure 3) This is due to the complexity involved in coding the bulk of the application logic and the difficulties associated with coordinating multiple independent software modules on disparate platforms. In contrast, the two-tier scheme allows the bulk of the application logic to be

developed in a higher-level language within the same tool used to create the user interface (Gallaugher, 1996).

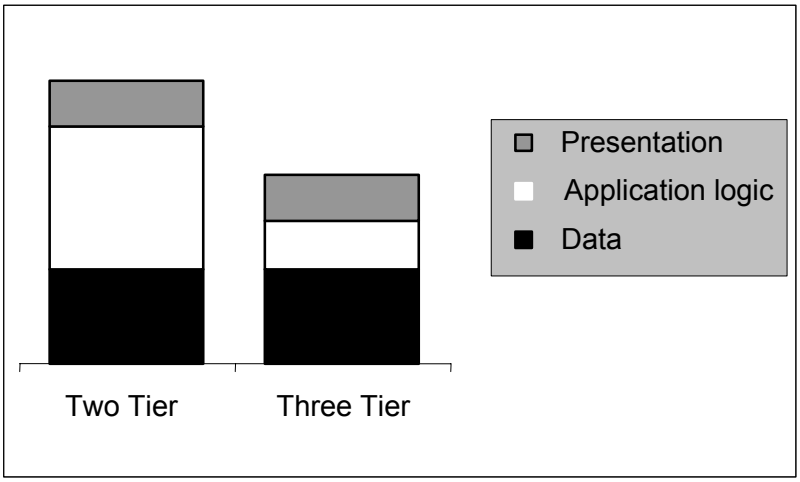

Figure 4. Comparison of Subsequent Development Efforts (From Gallaugher, 1996)

Subsequent development efforts may see three-tier applications deployed with greater speed than the two-tier systems. (Figure 4) This is entirely due to the amount of the middle-tier code, which can be reused from the previous applications (Gallaugher, 1996).

Table 1 shows a more detailed comparison of two-tier and three-tier applications (Edwards, 1999).

When the middle tier is divided in two or more units with different functions, the architecture is often defined as multi-tier (N-tier). A Web-based computing system is an example for a multi-tier architecture. In these systems, presentation and the presentation logic layers are implemented in client-side Web browsers using the content downloaded from a Web server. The presentation logic layer then connects to the application logic layer that runs on an application server, which subsequently connects to the database server(s) on the backside.

COMPARISON OF TWO-TIER AND THREE-TIER ARCHITECTURE			
No	**Property**	**Two-tier**	**Three-tier**
1	System administration	Complex (more logic on the client to manage)	Less complex (the application can be centrally managed on the server)
2	Security	Low (data-level security)	High (fine tuned at the service or method level)
3	Encapsulation of data	Low (data tables are exposed)	High (the client invokes services or methods)
4	Performance	Poor (many SQL statements are sent over the network; selected data must be downloaded for analysis on the client)	Good (only service requests and responses are sent between the client and server)
5	Scale	Poor (limited management of client communications links)	Excellent (concentrates on incoming sessions; can distribute loads across multiple servers)
6	Application reuse	Poor (monolithic application on client)	Excellent (can reuse services and objects)
7	Ease of development	High	Getting better
8	Server to server infrastructure	No	Yes (via server side middleware)
9	Legacy application integration	No	Yes (via gateways encapsulated by services or objects)
10	Internet support	Poor (internet bandwidth limitations make it harder to download fat clients and exacerbate the already noted limitations)	Excellent (thin clients are easier to download as applets or beans even as HTML pages; remote service invocations distribute the application load to the server)
11	Heterogeneous database support	No	Yes (three-tier applications can use multiple databases within the same business transactions)
12	Rich communication choices	No (only synchronous, connection-oriented calls)	Yes (supports connection oriented calls, connectionless messaging, queued delivery, publish-and-subscribe, and broadcast)
13	Hardware architecture flexibility	Limited (one has a client and a server)	Excellent (all three tiers may reside on different computers, or the second and third tiers may both reside on the same computer; with component-based environments, you can distribute the second tier across multiple servers as well)
14	Availability	Poor (cannot fail over a backup server)	Excellent (can restart the middle tier components on other servers)

Table 1. The Comparison of Two-Tier and Three-Tier Architectures (From Edwards, 1999)

D. CONCLUSION

In the year 2000, enterprise information systems require good performance, high security, reliability, less complex administrative tasks, and better application reuse.

Contrary to two-tier architecture, in the three-tier architecture only service requests and responses are sent between the client and server. Instead of sending many SQL statements and downloading the whole selected data for analysis, three-tier architecture offers better performance.

System security in the two-tier environment can be complicated since a user may require a separate password for each SQL server access. The proliferation of end-user query tools can also compromise database server security. But three-tier architecture enables us to develop more secure applications. Since the system is divided into parts, which can be developed separately, each component can use a unique protocol independent to other protocols. Even if unwanted users gains access to the Web server, they would not be able to get direct access to the database server and the confidential data. Also adding firewalls between the clients and the Web server will enhance the security.

In the three-tier architecture, the experts can develop and maintain each component (tier) separately. As the data server was developed and configured by the database administrators, the Web masters can configure Web server. This reduces the complexity of administrative tasks.

In the three-tier architecture, modularly designed middle-tier code components can be re-used by several applications. Reusable logic can reduce subsequent development efforts, minimize the maintenance workload, and decrease migration costs.

At this point, the three-tier architecture is the best model that satisfies the needs for an enterprise information system. At a functional level, the architecture of an enterprise information system can be made of four components: client, Internet firewall, Web server (application server) and, database server. (Figure 5)

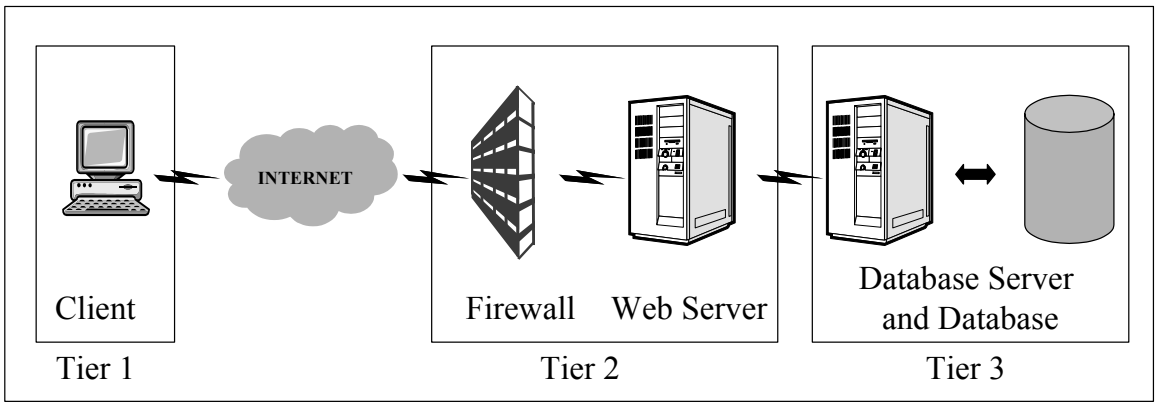

Figure 5. An Architecture Design Example for an Enterprise Information System (After Goodyear, 2000)

The remote clients connect to the Web server via the internet/intranet. The Web browser downloads the necessary client software to the client. The Web browser (first tier) sends a Web page request or data request to the Web server. The Web server works as a middle tier, and it takes the page request and ships the data request to the server extension program. Then server extension program accepts the requests and converts them to a form that the database server (third tier) can interpret. For the next step, the database server performs the specified task, such as query, insert or update and returns a result set to the server extension program. The server extension program converts the database result to a form that the Web browser can accept (e.g.: HTML), and finally it passes the result set to the Web server, which passes the final result to the Web browser. (Goodyear, 2000)

IV. EVALUATION OF SERVER-SIDE TECHNOLOGIES

This chapter defines the role of the server-side programs and the technologies used to create those programs. Later an evaluation of the current technologies will be presented.

A detail background on Web architecture and its components is available in Appendix B.

A. SERVER-SIDE TECHNOLOGIES

Many client requests can be satisfied by sending back pre-built documents, and these requests would be handled by the server itself. In many cases, however, a static result is not sufficient and a unique page/response must be generated for each request. Some of the reasons for this are

- The Web page is based on data submitted by the user, such as the order/confirmation page.
- The Web page is derived from data that changes frequently, such as new headlines generated on a daily or hourly basis.
- The Web page uses information from a database or other server-side sources.

In these cases, we must create the Web pages dynamically. At this point, server-side programs help us.

In this section we will study the technologies that enable us to create these programs. These programs run on a Web server, acting as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server. Their common responsibilities are

- **Read Any Data Sent by the User:** These data are usually entered in a form on a Web page, but could come from a Java applet or a custom HTTP client.

- **Look Up Any Other Information about the Request That Is Embedded in the HTTP Request:** Details about browser capabilities, cookies, the host name of the requesting client are all included in this information.
- **Generate the Results:** The process of generating the results may require talking to a database, executing a remote method invocation (RMI) or common object broker architecture (CORBA) call, invoking a legacy application, or computing the response directly.
- **Format the Results inside a Document:** This function mostly involves embedding the information inside an HTML page.
- **Set the Appropriate HTTP Response Parameters:** This tells the browser what type of document is being returned (e.g. HTML), setting cookies and caching parameters and other tasks.
- **Send the Document Back to the Client:** This document may be sent in text format (HTML), binary format (e.g. GIF images), or even in a compressed format like gzip that is layered on top of another underlying format.

Every program is not accessible to the Web server. In order to accomplish this, a program must have the following properties:

- The program should be able to be invoked by the Web server. When a request is issued from a user, the Web server must be able to locate and to execute the requested program.
- The Web server must be able to pass any form data to the program. When the Web server invokes the program, it needs a way to pass the HTTP request.
- Once the program is invoked there must be a standard entry point.

- After the program has processed the input data, it has to package the results and send them back to the Web server, which will, in turn, send them back to the Web browser.

The current popular server-side technologies are Common Gateway Interface (CGI), Web Server Application Programming Interfaces (API), Active Server Pages (ASP), Java Servlets and Java Server Pages (JSP). (Ayers, 2000)

1. Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is a standard way for interfacing external applications with Web servers. A plain HTML document that the Web server retrieves is static, which means it exists in a constant state: a text file that does not change. A CGI program, on the other hand, is executed in real-time so that it can output dynamic information. (Comer, 2001)

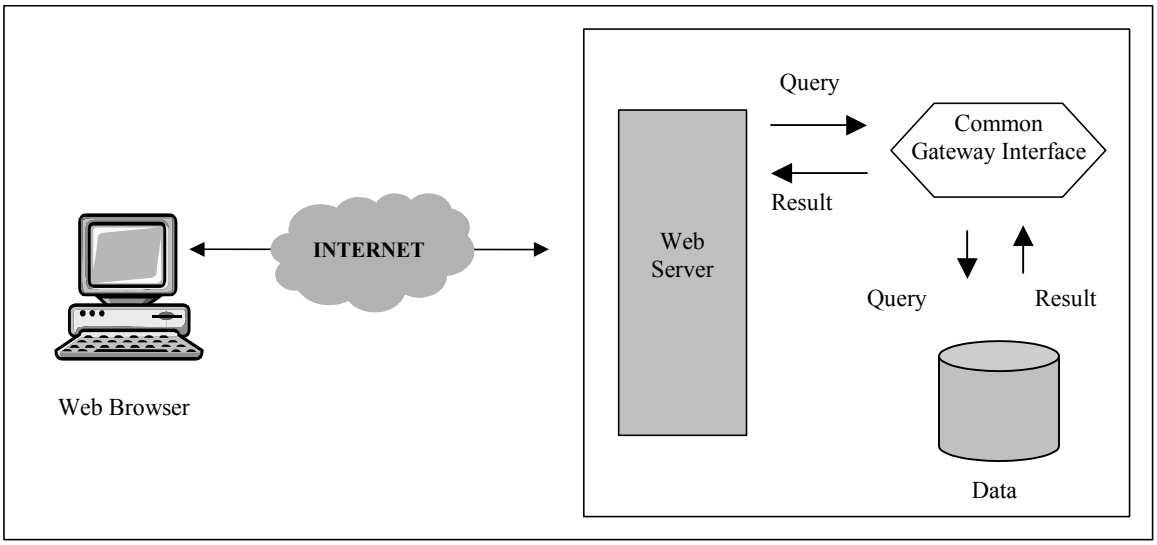

Figure 6. Database Connection with CGI

For an example, assume that we want to connect our database to the World Wide Web to allow people to query it. (Figure 6) Basically, we need to create a CGI program that the Web server will execute to transmit information to the database engine and receive the results back again and display them to the client. This is an example of a gateway where the CGI originated. (NCSA, 1999)

A CGI program can be written in any language that allows it to be executed on the system, such as C, C++, PERL, TCL, Unix shell, Visual Basic.

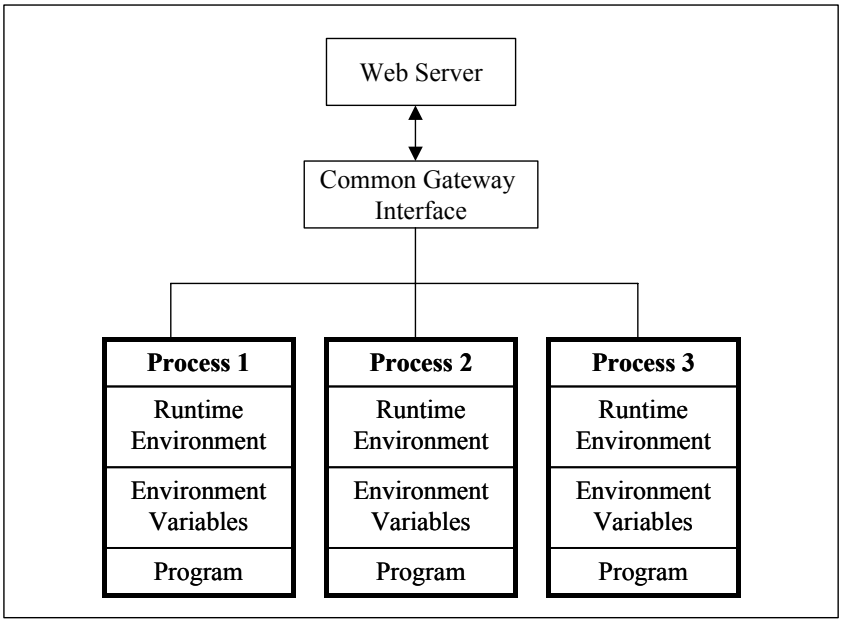

Figure 7. Creation of an Entire New Process for Each Request in CGI (From Ayers 2000)

The greatest disadvantage of the CGI is that for each client request, the Web server creates an entirely new process (Figure 7). Each process consists of its own set of environment variables, a separate instance of which runtime environment is required, a copy of the program and an allocation of memory for the program to use. It can be difficult to imagine what might happen to a server when a large number of requests are received simultaneously. This will certainly affect the respond speed and the performance of the hardware. (Ayers, 2000)

2. Web Server APIs (ISAPI, NSAPI)

Because of the inefficiencies of the CGI, some firms developed their own Application Programming Interfaces (API) to allow developers to write server side programs as shared libraries. Microsoft's Internet Server API (ISAPI) and Netscape's Netscape API (NSAPI) are well-known APIs. These libraries are designed to be loaded into the same process as the Web server and can service multiple requests without creating additional processes. (Ayers, 2000)

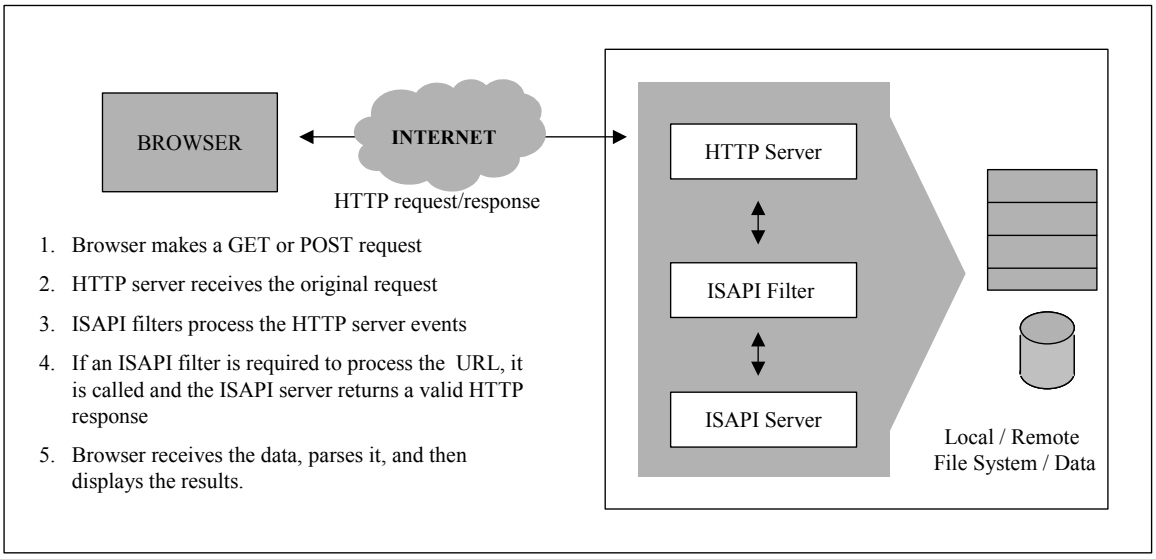

Figure 8. Request-Response Flow Diagram for APIs

Instead of creating new processes, the ISAPI server creates a thread pool when the server is initialized. Creating a thread take much less memory than creating a new process. A free thread from this thread pool serves the incoming connection. If the threads in the thread pool are all in use, the server can create additional threads to handle the waiting connections.

With ISAPI, the server calls the ISAPI DLL's entry point and leaves processing up to the extension or filter. Once processing is complete, the ISAPI DLL does any necessary cleanup and returns control of the thread to the server.

While this approach provides an efficient extension to the Web server, the APIs also have a few problems. (Ayers, 2000)

- **Platform Dependent:** Since these APIs are specific to a particular platform, any programs written using them can only be used on that platform. It would need extra tasks to move these programs into a different environment.
- **Thread Safe Requirement:** Since these libraries are accessed by multiple users simultaneously, they must be thread-safe. This means that they must be carefully developed about accessing global and static variables.

- **Access Violation Danger:** If a server program causes an access violation, since it is within the same process as the Web server, it has the potential of crashing the entire Web server.

3. Active Server Pages (ASP)

Another Web technology from Microsoft is Active Server Pages (ASP). ASP combines HTML, scripting, and server side components in one file with asp extension.

Such a file can be written using HTML, Jscript, and VBScript. The active server pages can also access the server components via the scripting. Such components can be written in any language as long as it presents a Microsoft's Component Object Model (COM). Executing everything on the server ensures that the pages are browser-independent, limited only by what the server can do.

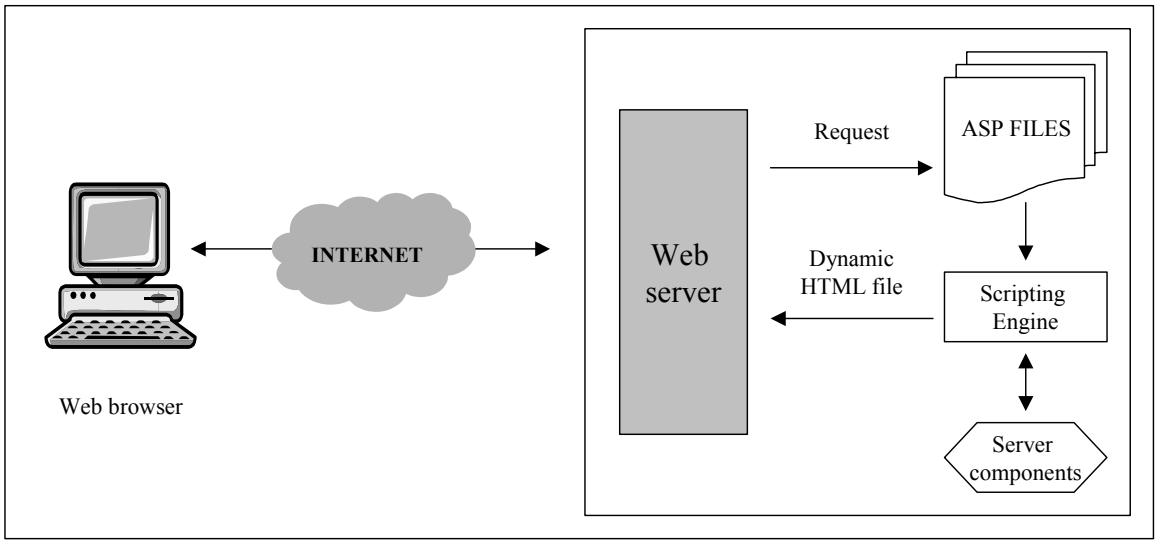

Figure 9. Request-Response Flow Diagram for ASP

The steps of running an ASP page can be summarized as follows: (Figure 9)

- The Browser sends a request for the ASP file to the Web server.
- The Web server recognizes the request for an ASP page.
- The Web server retrieves the ASP file from the disk or memory.
- The server sends the file to the ASP engine.
- The ASP page is processed from top to bottom as the scripting code encountered is executed and the result is produced as an HTML file.

- The resulting HTML file is sent back to the client browser.
- The HTML file including the result is interpreted by the Web browser and displayed.

One real disadvantage of Active Server Pages is that they can only be used with a Microsoft Web server, Internet Information Server (IIS) or Personal Web Server (PWS). While there are ports to other platforms and Web servers, the lack of COM support reduces their effectiveness. (Ayers, 2000)

4. Java Servlets and Java Server Pages

Servlets are Java technology's answer to Common Gateway Interface programming. A Java Servlet is a server-side program that services HTTP requests and returns results as HTTP responses.

Servlets are quite similar to Java applets. A servlet is a non-visual applet that runs on the Web server. It has a lifecycle similar to an applet and also runs inside a Java Virtual Machine (JVM). But servlets are for the server side, not for browsing, and they have extended functionality and more security features than an applet.

When a request for a specific servlet comes from a user, a server will use a different thread and then process the individual request. Since the threads use fewer resources, this gives servlets an advantage. Another benefit for the servlets is that because the servlet runs inside a JVM, it can easily be ported to other platforms supporting Java.

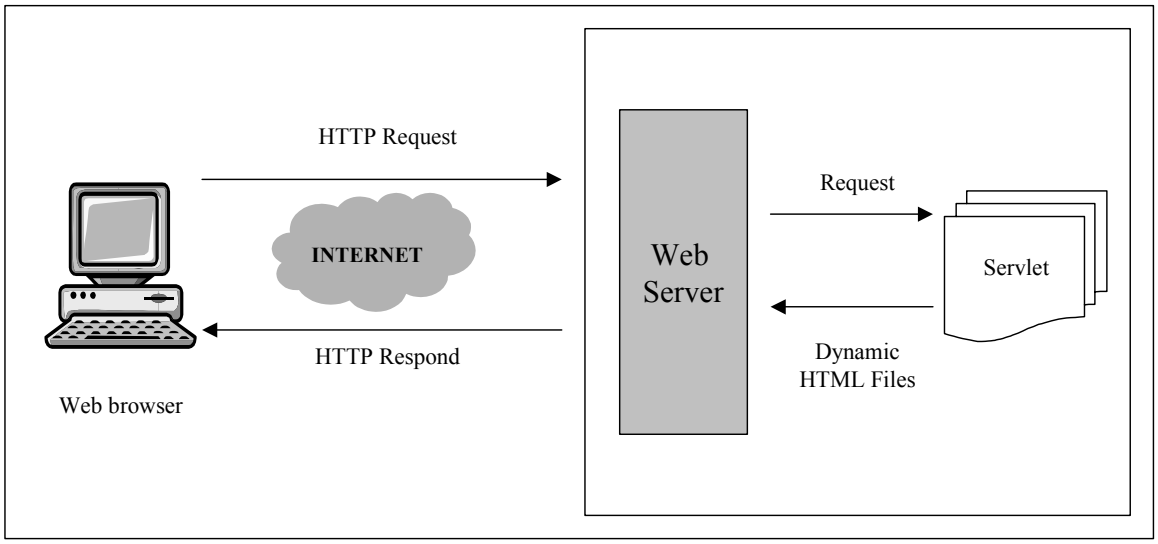

Figure 10. Request-Respond Flow Diagram for Java Servlets

When a user makes a request via HTTP, the Web server receives the request and forwards it to the servlet. If this is the first time the servlet will be used, the Web server will load the servlet into a Java Virtual Machine and execute it. The servlet receives the HTTP request, reads input parameters, and processes them accordingly. A response will be created and sent back to the server. The Server forwards the response to the client.

The Java Server Pages (JSP) is an extension of the Java Servlet technology. They are similar to Microsoft's Active Server Pages (ASP). A Java Server Page contains HTML, Java code and Java beans components. When a user requests a JSP file, a Web server will first generate a corresponding servlet, unless one already exists. The Web server then invokes the servlet and returns the resulting content to the user. JSPs provide a powerful and efficient dynamic page assembly mechanism that benefits from the advantages of the Java platform. (Ayers, 2000)

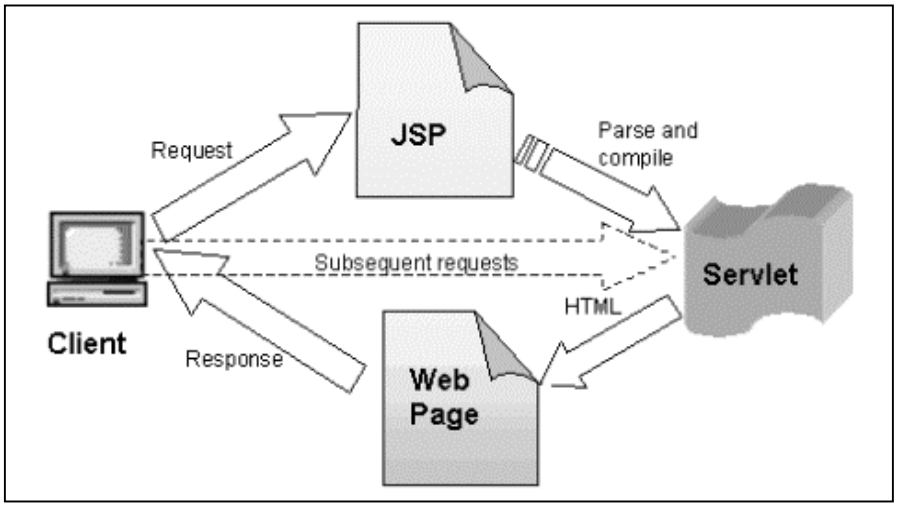

Figure 11. The First Call Initializes the JSP and Subsequent Requests Invoke Its Output.

By separating the page logic from its design and display, and by supporting a reusable component-based design, the JSP technology makes it faster and easier to build Web-based applications (Hall, 2001).

B. COMPARISONS OF THE TECHNOLOGIES

Until this point, each server-side technology was studied with its advantages and disadvantages. In this section we will compare these technologies from the view of security, performance, platform independence, reusability, and efficiency.

1. Security Issues

Security is an important issue in designing a Web-based application. The key security issues can be summarized as follows:

- Can the server run untrusted code or prevent it from accessing other data on the server?
- Is it possible to limit damage to the server caused by an application (such as wasting memory or causing a system crash)

As traditional CGI programs are often executed by general purpose operating system shells, programmers must be careful to filter out characters (such as back quotes and semicolons) that are treated specially by the shell. Another security vulnerability is

that some CGI programs and ASP files are processed by languages that do not automatically check bounds (such as array or string bounds or memory allocation/reallocation as in C and C++). This is a security hole for most of the system crashes. (Ayers, 2000)

For servlets, the Java Virtual Machine (JVM) can restrict the environment available to the servlet, so it cannot access other critical data from the Web server. Also, JVM provides a greater degree of control over application behavior than the other approaches because most of the memory-checking protection features and garbage collector are central parts of the Java programming language. This fills the gap against system crashes. (Hall, 2001)

2. Performance

With traditional CGI, a new process is started for each HTTP request. This means for "N" number of requests there will be again "N" number of CGI programs running in the memory. Also when a CGI program finishes handling a request, the program terminates. This makes it difficult to cache computations and to keep database connections open. If a communication with a database is needed, for each CGI request, a new connection is established. (Sun, 2000)

Since servlets are developed using Java, they provide superior performance compared to ASP, which uses interpreted VBscript or Jscript languages. With servlets, the Java Virtual Machine keeps running and handles each request using a lightweight Java thread, not a heavyweight operating system process as in CGI. As the servlets remain in memory even after they complete a response, it is straightforward to store arbitrarily complex data between requests, and the connections can be kept open. As a result Java servlets use less system resources, scale better and provide improved performance.

3. Platform Independence

ASP and CGI are basically Microsoft technologies, and they rely on Microsoft, and APIs are dependent to their Web server manufacturer. Even though there are some commercial off-the-shelf (COTS) products on the market, they require extra software support. CGI is supported by different Web servers, but because it uses shell commands,

a CGI program that uses Unix shell command, must be redesigned if one wants to move it to a Windows machine.

In contrast Java servlets and Java server pages like their predecessor, Java programming language, are designed to be platform and server independent. One can easily transfer one's application from one platform to another. (Hall, 2001)

4. Reusability and Ease of Maintenance

Code reuse is an important feature of software engineering and programming (Pressman, 2001). For most developers, it is easier to read someone else's Java code and determine what that person has done than to read a Perl script. Java portable application components (utility classes or Java Beans) that perform specific tasks can be reused. Developers can create customized JSP tag libraries to distribute reusable functions to page authors.

Time spent maintaining an application contributes to its ongoing cost. Differences exist between Perl applications and the Java Servlet technology-based solutions. These differences are due to the nature of the Java language. As an inherently object-oriented language, Java is simpler to maintain than Perl and ASP (as a mixture of scripting languages and HTML) (Hall, 2001).

D. CONCLUSION

At this point we must ask, "What kind of a technology are we looking for? And what are the features that we want from a server-side program?" From this perspective, one determines that we need

- A portable technology that provides the flexibility to change the application environment easily.
- A reliable language that provides us trust and robustness,
- A technology that is cost effective,
- A technology that provides the needed level of security,
- A technology that provides the needed performance.

As Java is built on the principle of "build once, run everywhere," therefore it can overcome cross-platform obstacles. A future change in the configuration of the architecture or system environment can affect the application only at a minimum level. Thus Java is portable. (Hall, 2001)

Java emphasizes early checking for possible problems, later dynamic checking and eliminates situations that are error-prone. In other languages, while many problems would show up at runtime, such problems are detected at compile time by the Java compiler. This makes Java reliable. (Ayers, 2000)

Tools to develop a Java servlet application can be downloaded from the Sun homepage at no cost. Even though environments and products (Web server, database server) support Java as configured, adding Java-servlet support costs little. This is in contrast to many of the other competitors, such as Microsoft ASP, which require an initial investment to purchase a proprietary package. To increase the portability of these tools, one needs extra tools. Chilisoft is one example of a company offering such solutions for Microsoft ASP. But this also increases the cost. Therefore Java is cost effective.

As there are no shell commands in Java servlets and the unique error and security protection features (such as memory protection) are a central part of the Java programming language itself, Java ensures that a programmer cannot write subversive code for applications. This makes Java secure. (Ayers,2000)

By using threads, instead of system processes for each request, Java servlets use less system resources, scale better and provide improved performance. So Java is efficient. (Hall, 2001)

Under these circumstances, Java servlets and JSP appear to be the best choice for a Web-based application.

V. RELATIONAL DATABASES AND THE STRUCTURED QUERY LANGUAGE

This chapter provides a brief overview of Relational Databases and Structured Query Language (SQL).

A. RELATIONAL DATABASES

1. Database

A database is a collection of data, describing the activities of one or more related organizations (Ramakrishnan, 2000). Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. A database is restricted to some implicit properties; it represents real world aspect, consists of a coherent collection of data that has inherent meaning, and is designed, built and populated for a specific purpose (Elmasri, 1994). For example, the data that are related to the inventory of a Naval Supply Center could be stored in a database.

2. Database Management System

Database Management System or DBMS is an application, designed to assist in maintaining and using large collections of data (Ramakrishnan, 2000). A DBMS enables the user to perform the following operations:

- **Define a Database:** This involves specifying data types, structures, and constraints for the data to be stored in the database.
- **Construct a Database:** This is the process of storing the data on a storage medium (such as magnetic disk, tape or optical disk) controlled by the DBMS.
- **Manipulate a Database:** This includes functions such as querying the database, updating the database and generating reports from the data.

3. DBMS vs. File Systems

Database technology involved overcoming the limitations of traditional flat files. In a flat file system, each user designs and implements the files needed for a specific application. In a DBMS, the database forms a single repository that is defined once and accessed by multiple users, each of whom may see a different view of the database. The data must also be restored to a consistent state if the system crashes while changes are being made. Neither the users nor the programmers need to worry about how data are physically stored in the database.

By storing the data in a DBMS, rather than as a collection of operating system files, we can use the DBMS features to manage the data in a robust and efficient manner (Ramakrishnan, 2000).

A DBMS provides several advantages by

- Reducing data duplication,
- Restricting authorized access,
- Defining and enforcing integrity constraints,
- Providing tools for backup and recovery,
- Enabling the database administrator to enforce standards,
- Reducing the application-development time,
- Permitting changes to the structure as the requirements change,
- Making the data available to the authorized users in an up-to-date state (Elmasri, 1994).

4. Data Models and Relational Data Model

A data model is a collection of high-level data description constructs that hide many low-level storage details. A DBMS allows a user to define data to be stored in terms of a data model (Ramakrishnan, 2000).

Database engineers define five different types of databases:

1. Relational,

2. Hierarchical,
3. Network,
4. Object-oriented
5. Object-relational model.

A popular type in use today is the relational model (Ramakrishnan, 2000). This is because they are built on a provable mathematical foundation that hastened the process of developing a generalized query language (Taylor, 1999). Therefore the focus of this chapter and the implementation of our prototype will be the relational database model.

Elmasri (1994) defines a data model, as a set of concepts that can be used to describe the structure of the database. From this perspective, a relational-data model represents the database as a collection of *relations,* which can be thought of as a set of *records.* Each relation is defined as a "table of values." Within a table, each row is called a *tuple* and represents a unique value in the database.

In the relational model each table must have a unique column called a *primary key* that is used to identify tuples in the relation. Any column that is part of a primary key cannot be null. This rule is referred to as *entity integrity.* The following table illustrates a sample relational database table of a hypothetical bookstore to track books.

BOOKS					
ISBN	Title	Author	Publisher_name	Pub_date	Price
012-23001-34	Database Management Sys.	Ramakrishnan	McGraw Hill	2000	$90.00
023-23002-65	JDBC and Java	Reese	O'Reilly	2001	$45.00
982-30030-87	XML databases	Williams	McGraw Hill	2000	$65.00

Table 2. Example Database Table for Books

In Table 2, each column represents a different attribute of a book. Each row represents a different book. An ISBN attribute is unique for each book. Therefore it is chosen as a primary key.

Foreign keys are used to define relationships between tables. A foreign key in one table is a reference to a primary key in another table. A row that refers to another row

must refer to an existing row in that relation. This rule is referred as *referential integrity* (Ramakrishnan, 2000). Now consider the following table.

PUBLISHERS		
Publisher_name	Phone	Pub_city
McGraw Hill	407-3567484	Los Angeles
O'Reilly	888-9082733	Chicago

Table 3. Example Database Table for Publishers

In the BOOKS table, the foreign key was Publisher_name, which refers to a unique row of data in the PUBLISHERS table. Publisher_name is the primary key in the PUBLISHERS table.

A relationship can be one of these forms: One-to-One, One-to-Many, and Many-to-many. "One-to-one" means that for every occurrence of an instant of data in a table, only one instance will occur in another. "One-to-Many" means that for every occurrence of an instant of data in a table, many instances of data may occur in the other table (Ramakrishnan, 2000). For example, if we look at the relationship between the PUBLISHERS and the BOOKS, we will see that for every Publisher_name in PUBLISHERS, there may be several books in the BOOKS table.

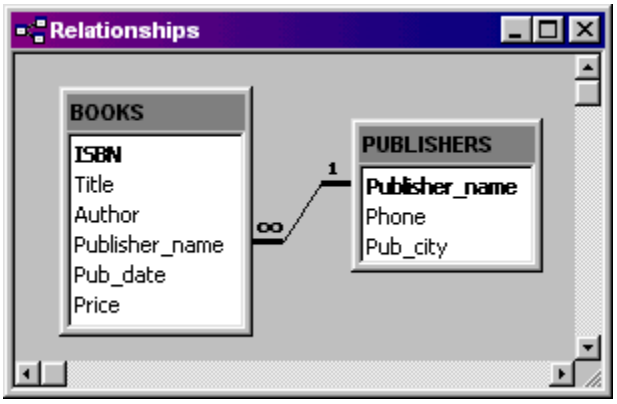

Figure 12. Relationships Diagram for Sample Tables

Figure 12 shows the relationship diagram for the sample tables created in MS Access. These diagrams are used to describe the relations of the tables visually. In this

figure, we can see the primary keys of each table and the relationship between them (many-to-one).

B. STRUCTURED QUERY LANGUAGE

Structured Query Language (SQL) (pronounced sequel) was originally developed at IBM in the SEQUEL-XRM and System-R projects in 1977. Almost immediately, other vendors introduced DBMS products based on SQL. This made the SQL a standard for relational database industry (Ramakrishnan, 2000).

SQL is a database language used to extract data from relational databases. Relational database users quickly learn this language because it expresses its logic with English-like syntax. SQL consists of a set of standard commands that can be understood by all compliant Relational Database Management Systems (RDMS).

1. SQL Statements/Commands

SQL commands are divided into categories, Data Definition Language (DDL) statements and Data Manipulation Language (DML) statements (White, 1999).

a. Data Definition Language (DDL)

This subset of SQL supports the creation, deletion, and modification of definitions for tables. The DDL also provides commands to specify access rights or privileges to tables or views. The following tables show the common commands to create or modify tables and other database objects (White, 1999).

SQL Command	Description
CREATE TABLE	Create a table with the name of the table and columns provided by the user.
DROP TABLE	Delete a table definition and all rows in the table.
ALTER TABLE	Change the definition of a table. It is also possible to add or drop a column, change a column definition or add/remove constraints defined for the table.

Table 4. A List of the Most Used SQL Commands of DDL

SQL Command	Syntax	Example
CREATE TABLE	**CREATE TABLE** <tablename> (col-def, …., col-def, tab-constr, …tab-constr);	To create the BOOKS table: **CREATE TABLE** BOOKS (ISBN **number**(5) **primary key**, Title **char** (30), Author **char**(30), Publisher_name **char** (30), Pub_date **date**, Price **number**(3,2));
DROP TABLE	**DROP TABLE <**tablename>	To delete the whole BOOKS table: **DROP TABLE** BOOKS;
ALTER TABLE	**ALTER TABLE <**tablename>	To add a new column to the BOOKS table as book Book_edition: **ALTER TABLE** BOOKS **ADD** Book_edition **char(20);**

Table 5. Examples on the Most Used SQL Commands of DDL (The Examples are Implemented on the Sample Tables of BOOKS and PUBLISHERS.)

b. Data Manipulation Language (DML)

This subset of SQL allows users to pose queries and to insert, delete, and modify rows. The following tables shows the common commands for these purposes (Reese, 2000).

SQL Command	Description
SELECT	This is the most frequently used SQL command and is used to query the database and to display selected data to the user. It mostly consists of three main clauses: SELECT, FROM, and WHERE. Normally SELECT keyword is followed by a list of columns. FROM keyword is followed by a list of tables and WHERE keyword is followed by a list of search conditions.
INSERT	Used to add a row into a table that already exists.
UPDATE	Used to modify a record or more of data that meets a specified condition.
DELETE	Used to delete a row or more of data that meet the condition specified in the WHERE clause.

Table 6. A List of the Most Used SQL Commands of DML

SQL Command	Syntax	Example
SELECT	**SELECT** <expression> **FROM** <TABLENAME> **WHERE** <search condition>;	To list the name and the author of the books published by the "McGraw Hill": **SELECT** Title, Author **FROM** BOOKS **WHERE** Publisher_name = 'McGraw Hill';
INSERT	**INSERT INTO** <tablename> [(column,…column)] **VALUES** (expression, …expression);	To add a new publisher to the publishers to the PUBLISHERS table: **INSERT INTO** PUBLISHERS (Publiher_name,Phone,Pub_city) **VALUES** ('Wrox', '800-786453', 'New York';
UPDATE	**UPDATE** <tablename> **SET** <column> = <expression> [**WHERE** <search-condition>];	To change the price for the book with ISBN "012-23001-34": **UPDATE** BOOKS **SET** Price = '$35' **WHERE** ISBN = '012-23001-34';
DELETE	**DELETE FROM** <tablename> [**WHERE** <search-condition>];	To delete the book record from the BOOKS with the name "XML databases": **DELETE FROM** BOOKS **WHERE** Title = 'XML databases';

Table 7. Examples on the Most Used SQL Commands of DML (The Examples are Implemented on the Sample Tables of BOOKS and PUBLISHERS.)

2. How SQL is Used

There are a variety of ways to use SQL to define and manipulate data in a Database Management System (Ramakrishnan, 2000).

One way to define and manipulate data in a DBMS is to embed SQL statements in a high level programming language. This is called as Embedded SQL (ESQL). Embedded SQL allows programmers to place SQL statements into the host language. These statements are identified within the host language by marking their starting and ending points with special delimiters. When compiling a program with embedded SQL statements, a precompiler translates these statements into equivalent host language source

code. After this precompiling, the host language compiler compiles the resulting source code (Ramakrishnan, 2000).

Another way to execute SQL statements is called Stored Procedures. A stored procedure is a predefined and compiled procedure that is stored in the database and can be used by the client's applications and can be called by name (Davidson, 2001).

Stored procedures have many benefits. For example, once they are executed for the first time, there is no need to parse, optimize, or compile these stored procedures again. For this reason, stored procedures are fast in client/server environment. The stored procedures contain multiple queries, but to execute these stored procedures only one line is necessary. The parameters are passed to a stored procedure enabling different clients to access the same stored procedure.

Sometimes instead of executing predefined SQL queries, we can need to obtain the query from the user at the run time. Such an application is also available with SQL. This is called Dynamic SQL. Dynamic SQL features allow a query to be constructed (and executed) at the run time (Ramakrishnan, 2000).

Another alternative to embedded SQL is to use a callable SQL Application Programming Interface (API) for database access. An API does not require a precompiler to convert SQL statements into a high-level language, which can then be compiled and executed on the database. While embedded SQL is DBMS-independent only at the source level, applications using SQL APIs are DBMS-independent both at the source code and executable level. The SQL Access Group Call Level Interface (SAG CLI) specifies a common API for accessing multiple databases (Ramakrishnan, 2000). Open Database Connectivity (OBDC) and Java Database Connectivity (JDBC) are some of the well-known APIs for accessing multiple databases. APIs and JDBC will be explained in Chapter V.

C. SUMMARY

A Database Management System is software that supports the management of large collections of data. Such a system provides efficient data access, data independence,

data integrity, security, support for concurrent access, and recovery from system failures. Compared to using operating system files, using a DBMS has many advantages.

SQL is a structured query language that enables high-level access and modification of the data. SQL is easy to learn and to use. The SQL can be used directly on top of a database, embedded in a high level language or predefined and compiled procedures stored in the database.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SQL APPLICATION PROGRAMMING INTERFACES AND JAVA DATABASE CONNECTIVITY

This chapter gives a brief explanation of SQL Application Programming Interfaces (API) and current APIs. Later a more detailed overview on Java Database Connectivity is provided.

A. SQL APPLICATION PROGRAMMING INTERFACES

In Chapter IV, it was stated that another alternative to Embedded SQL was to use the callable SQL Application Programming Interface (API) for database access. In contrast to embedded SQL, SQL APIs allow a single executable to access different DBMS without recompilation (Ramakrishnan, 2000).

To access SQL from applications, the SQL Access Group (SAG) developed a standard Call-Level Interface (CLI). The CLI describes how SQL statements are issued from a programming language and how the results are bound to program variables.

The SAG CLI specifies a common API for accessing multiple databases. SAG APIs are based on dynamic SQL and support; Database connection requests through a local driver, SQL preparation requests, execution requests, statement termination requests, connection termination requests. SAG merged with X/open in 1995 and still continues its activities as X/open SQL Access Group.

Open Database Connectivity (ODBC) is a data-access standard developed by the Microsoft Corporation as an extended version of the SAG CLI. In addition to SAG CLI's basic functionality, ODBC provides methods to retrieve information about the database and to handle multimedia data types, and can connect to multiple kinds of databases on different platforms. But as the applications created using ODBC will not be portable to other platforms without code modifications and as the changes in the driver software API may require recompilation of code, ODBC has some disadvantages. Because of these drawbacks of ODBC, JDBC was developed by Sun Microsystems as a low-level API for invoking SQL commands directly on different vendor databases.

Later Microsoft introduced new APIs beyond ODBC such as OLE (Object Linking and embedding) DB, ADO (ActiveX Data Objects), RDS (Remote Data Service), and UDA (Universal Data Access). Still the current version of JDBC, JDBC 2.0 API contains extra features not found in these APIs (White, 1999).

B. JAVA DATABASE CONNECTIVITY

The JDBC is a high-level Application Programming Interface (API) for accessing any kind of tabular data. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of Relational DBMSs. JDBC abstracts the vendor specific details and generalizes the most common database access functions. The result is a set of methods formerly contained within the Java.sql and Java.text packages. This enables the developer to write database applications using a pure Java API. These methods can be used on any database providing JDBC connectivity through a vendor specific JDBC driver. If the developer needs a switch on a database, the written code can be kept as it is, one need only change the database driver (Ayers, 2000).

A JDBC driver is what the Java Virtual Machine (JVM) uses to translate the generalized JDBC calls into the vendor specific database calls that the database will understand. These drivers are Java classes and are loaded at runtime. This allows for maximum flexibility when deploying applications that may access databases from multiple vendors. JDBC drivers exist for nearly most of the databases. Typically the database vendor supplies the JDBC driver, but third party JDBC drivers can also be used.

1. JDBC Driver Types

Based on the architectural relationship between the application and the data source, JDBC drivers are classified into four types: (Ramakrishnan, 2000).

- Type 1: Bridges
- Type 2: Direct Translation to the Native API
- Type 3: Network Bridges
- Type 4: Direct Translation over Sockets

a. Type 1: Bridges

This type of driver translates JDBC function calls, into function calls of another API that is not native to the DBMS. An example is a JDBC-ODBC bridge. In this case, the application loads only one driver, namely the bridge. For instance, a Microsoft Access database can only be accessed using the JDBC-ODBC Bridge.

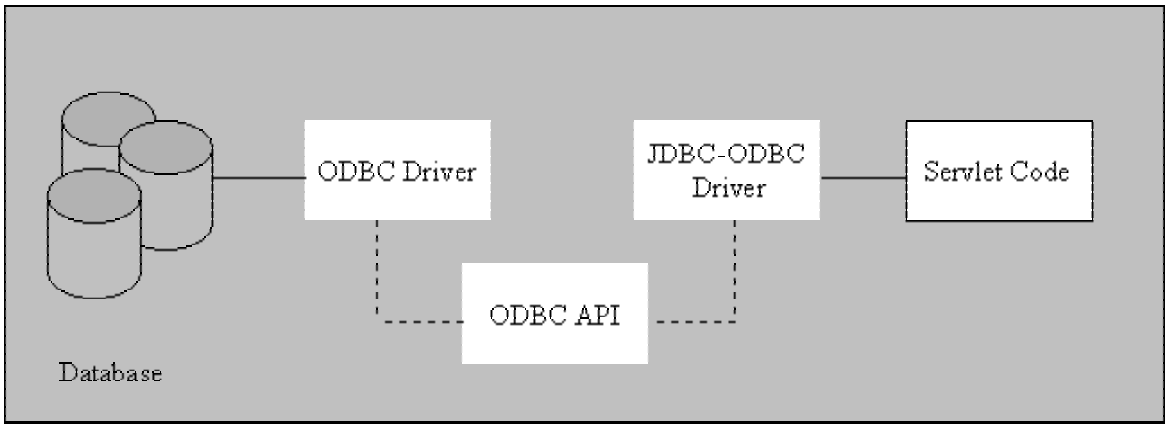

Figure 13. An Example Architecture Using Type 1 JDBC Driver (From Ayers, 2000)

According to Ayers (2000), accessing ODBC data sources using JDBC is not a very efficient solution. Not only must the system pass the database call through multiple layers, but it also limits the functionality of the JDBC code wherever the ODBC driver can handle this operation.

b. Type 2: Direct Translation to the Native API

This driver translates JDBC function calls directly to method invocations of the API of one specific data source. It is similar to the first type, but here it has one less layer (no ODBC translation layer is needed). The driver is dynamically linked and is specific to the data source. The database will translate the request and send its results back through the API, which will forward them back to the JDBC driver. The JDBC driver will format the results to conform to the JDBC standard and return them to the program.

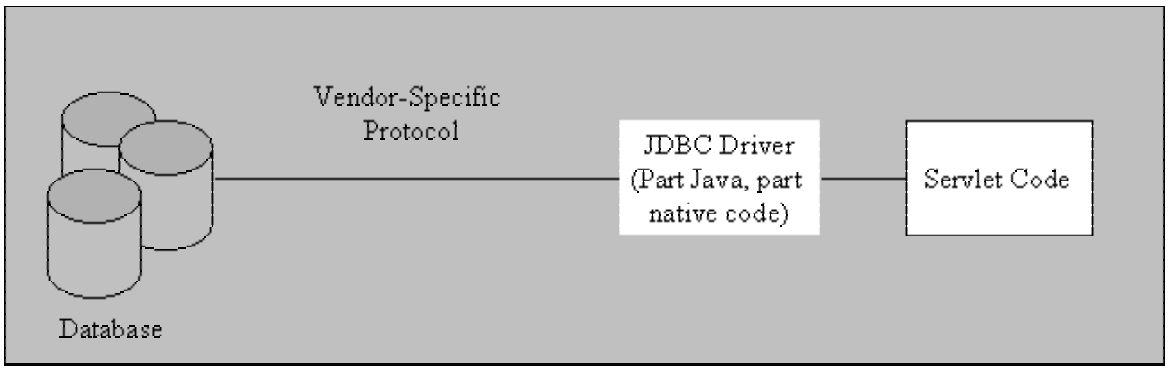

Figure 14. An Example Architecture Using Type 2 JDBC Driver (From Ayers, 2000)

c. Type 3. Network Bridges

The driver talks over a network to a middle-ware server that translates the JDBC requests into DBMS-specific method invocations. The Java program sends a JDBC call through the JDBC driver and straight to the middle tier without translation. The middle tier then handles the request using another JDBC driver to complete the request. The only problem with this is that the middle tier may be using a Type 1 or Type 2 drivers to talk to the database (Ayers, 2000).

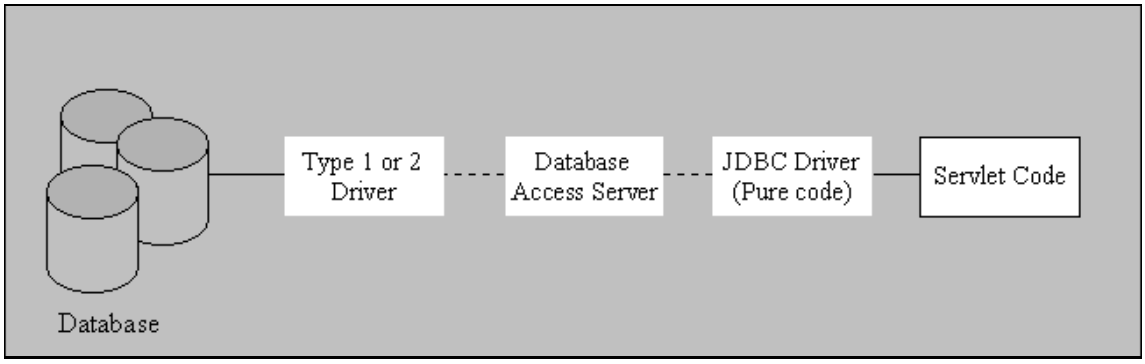

Figure 15. An Example Architecture Using Type 3 JDBC Driver (From Ayers, 2000)

d. Type 4: Direct Translation over Sockets

Here, instead of calling the DBMS API directly, the driver communicates directly with the DBMS through sockets. According to Ayers (2000), this is the most

efficient method of accessing a database. All of the major database vendors provide Type 4 JDBC drivers for their databases.

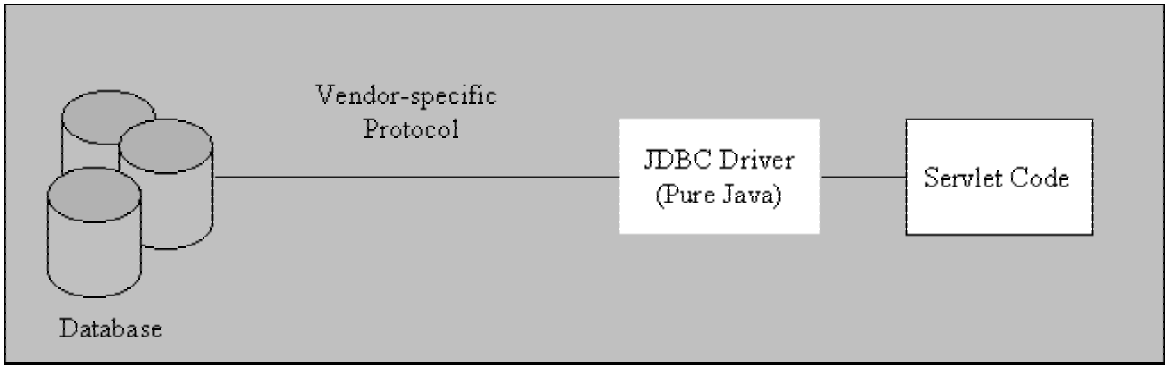

Figure 16. An Example Architecture Using Type 4 JDBC Driver (From Ayers, 2000)

White (1999) summarizes these four types in a simple table. The table uses the following definitions for types of network connections

- "Direct" is a connection that a JDBC client makes directly to the DBMS server, which may be remote.
- "Indirect" is a connection that a JDBC client makes to a middleware process that acts as a bridge to the DBMS server.

DRIVER CATEGORY	ALL JAVA	NETWORK CONNECTION
Type 1	No	Direct
Type 2	No	Direct
Type 3	Client Yes Server Maybe	Indirect
Type 4	Yes	Direct

Table 8. JDBC Driver Categories

2. JDBC API

JDBC API is composed of the classes and interfaces found in the Java.sql and Java.text packages. The following figure shows all the classes and the interfaces of the JDBC 2.1.

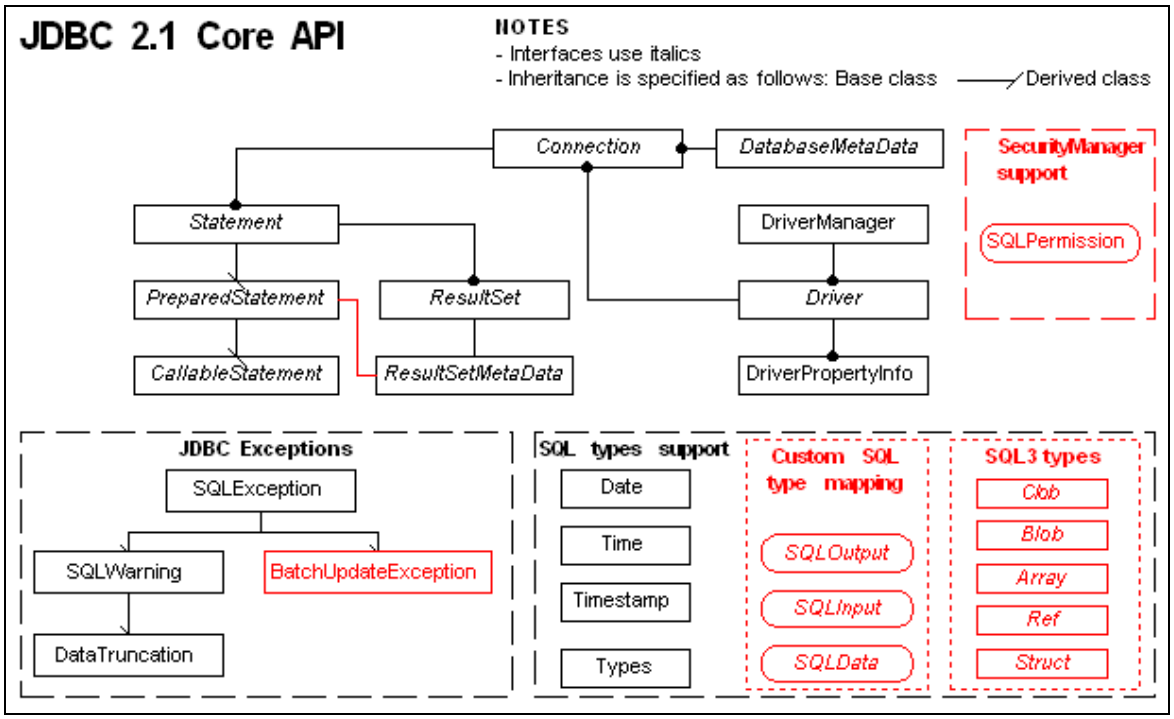

Figure 17. The Classes and Interfaces of JDBC 2.1

According to Whitten & Hapner (1999) the most important classes and interfaces are the DriverManager class, Connection interface, Statement interface, PreparedStatement interface, CallableStatement interface, and ResultSet interface. The following figure shows these classes and their relations.

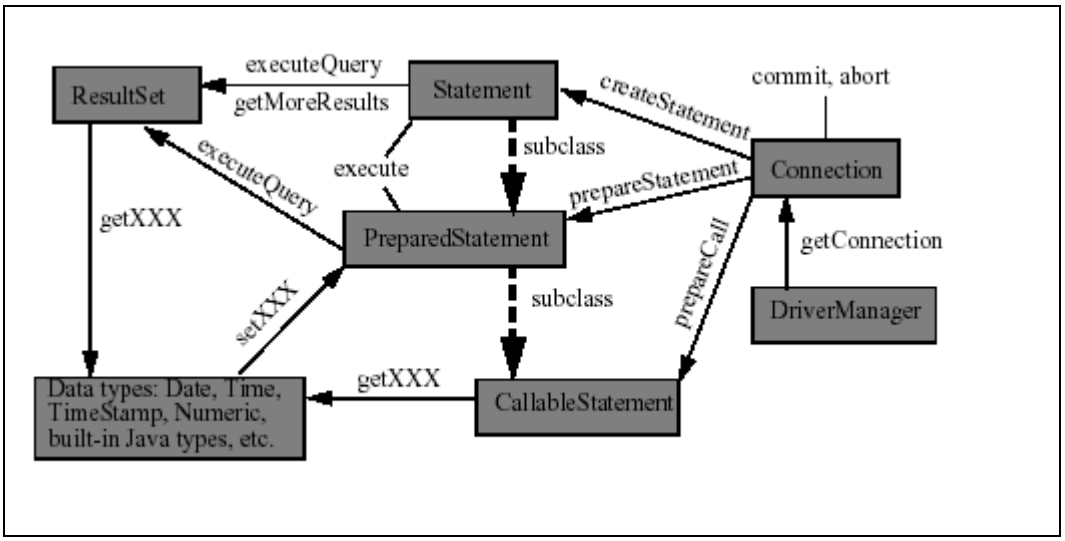

Figure 18. The Relations of Most Important Classes and Interfaces of JDBC 2.1 API (From White & Hapner, 1999)

These classes and interfaces will be discussed in the following sections.

a. Driver Manager

The DriverManager class is the traditional management layer of JDBC, working between the user and the drivers. It keeps track of the drivers that are available and handles establishing a connection between the database and the appropriate driver (White, 1999). Keeping track of the drivers is done by the system property called "jdbc.drivers". Creation of the connections are done by invoking getConnection() method on this interface.

b. Connection Interface

A connection object represents a session with a database. It provides an application with Statement objects (and its subclasses) for that session. The SQL statements that are executed, as well as the results that are returned over that connection, are included in this object. A single application can have one or more connections with a single database, or it can have connections with many different databases (White, 1999).

c. Statement Interface

This class represents an embedded SQL statement, which an application uses to perform database access (Reese, 2000). There are actually three kinds of Statement objects; Statement, PreparedStatement and CallableStatements. A Statement object is used to execute a simple SQL statement without any parameters, a PreparedStatement object is used to execute a precompiled SQL statement with or without parameters, CallableStatement object is used to execute a call to a database stored procedures (White, 1999).

d. ResultSet Interface

A ResultSet object represents a database result set table, generated by executing a SQL statement. It provides an application with access to database queries one row at a time (Reese, 2000). A ResultSet maintains a cursor pointing to its current row of data being manipulated. The application then moves through results sequentially until all results have been processed or the ResultSet closed.

C. BASIC STEPS IN USING JDBC

There are seven standard steps in querying databases (Hall, 2001):

1. Load the JDBC driver.
2. Define the connection URL.
3. Establish the connection.
4. Create a statement object.
5. Execute a query or update.
6. Process the results
7. Close the connection.

The following sections will discuss these steps and provide simple examples.

1. Load the Driver

To communicate with the database using JDBC, first a connection must be established through the appropriate loaded driver. To load the driver, the JDBC driver can be specified using the jdbc.drivers system property, or it can be set at runtime using the Class.forname(String driverName) method. The following is a simple example

```
try {
    Class.forName("Oracle.jdbc.driver.OracleDriver";
} catch (ClassNotFoundException cnfe) {
    System.err.println("There is an error loading driver: " + cnfe);
}
```

Figure 19. A Simple Example Loading a JDBC Driver

In this example we are loading the OracleDriver for our connection. As can be seen from the figure, Class.forname method, throws ClassNotFoundException if the driver cannot be found.

2. Define the Connection URL

Once the JDBC driver is loaded, the location of the database server must be specified. URLs referring to the databases use the jdbc:protocol and have the server host,

port, and database name embedded with the URL (Hall, 2001). The following is a simple example for defining a URL.

```
String host = “dbhost.mycompany.com”;
String dbName = “bookStore”;
Int port = 1234;
String OracleUrl = “jdbc:oracle:thin:@” + host + “:” + port + “:” + dbName;
```

Figure 20. A Simple Example for Defining a URL

If a jdbc-odbc bridge were used, our URL would be “jdbc:odbc:bookstore”

3. Establish the Connection

To make the connection, all we need is to pass the URL, the database user-name, and the password to the getConnection method of the DiverManager class.

```
String userName = “System”;
String password = “Manager”;
try {
Connection connectionOne =
        DriverManager.getConnection (OracleUrl, userName, password);
} catch (SQLException sqle) {
    System.err.println(“There is an error creating connection: “ + sqle);
}
```

Figure 21. Simple Example for Establishing a Connection

As we see from the example we need to catch the SQLException. As the SQLException is also thrown by the createStament and executeQuery methods, the try blocks will not be repeated for the next examples.

By using the getMetaData method of connection, information about the database can also be accessed.

4. Create a Statement

A statement object is used to send queries and commands to the database. To create a statement, createStatement of the Connection will be used.

```
Statement statement = connection.createStatement( );
```

Figure 22. Creating a Statement Object

5. Execute a Query or Update

Once a Statement object is created, SQL queries can be sent by using the executeQuery method , which returns an object of type ResultSet.

```
String queryOne =
" SELECT Book_name, Author_name FROM Books WHERE Publisher_name = 'Wrox' " ;
ResultSet resultOne = statement.executeQuery(queryOne);
```

Figure 23. Executing a Query

Instead of querying the database, we could modify it by using the executeUpdate method. In that case, no results would be sent back.

6. Process the Results

In the previous example, the result of our query was sent back as a ResultSet object. To handle the results, ResultSet provides us multiple methods. One of them is the next() method. By using this method, we can move the cursor from one row to the next row. Within a row, by using the getXXX method that takes an index or column name as an argument, the results can be gotten as a variety of different Java types (Hall, 2001).

```
System.out.println ("Book Name          Author Name");
System.out.println ("--------------      -----------------");
While (resultOne.next( )){
   String bookName = resultOne.getString("Book_name");
   String authorName = resultOne.getString("Author_name");
   System.out.println(bookName + "     " + authorName);
}
```

Figure 24. Processing the Results

7. Close the Connection

Closing the connection requires only connection.close( ) method. But this step should be postponed if additional database operations are expected because opening a connection usually has a large overhead. (Hall, 2001).

D. SUMMARY

JDBC is a low-level API that establishes a connection with a database, sends SQL statements, and processes the results. It overcomes some of the drawbacks of the ODBC.

As Java Servlets were selected as the server-side programming technology, JDBC allows us to use pure Java code throughout our applications.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. DESIGN AND IMPLEMENTATION

Pressman (2001) suggests the "spiral process model," as a development strategy for Web-based applications. The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.

Using the spiral model, software is developed in a series of incremental releases. During early iterations, the incremental release might be a prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

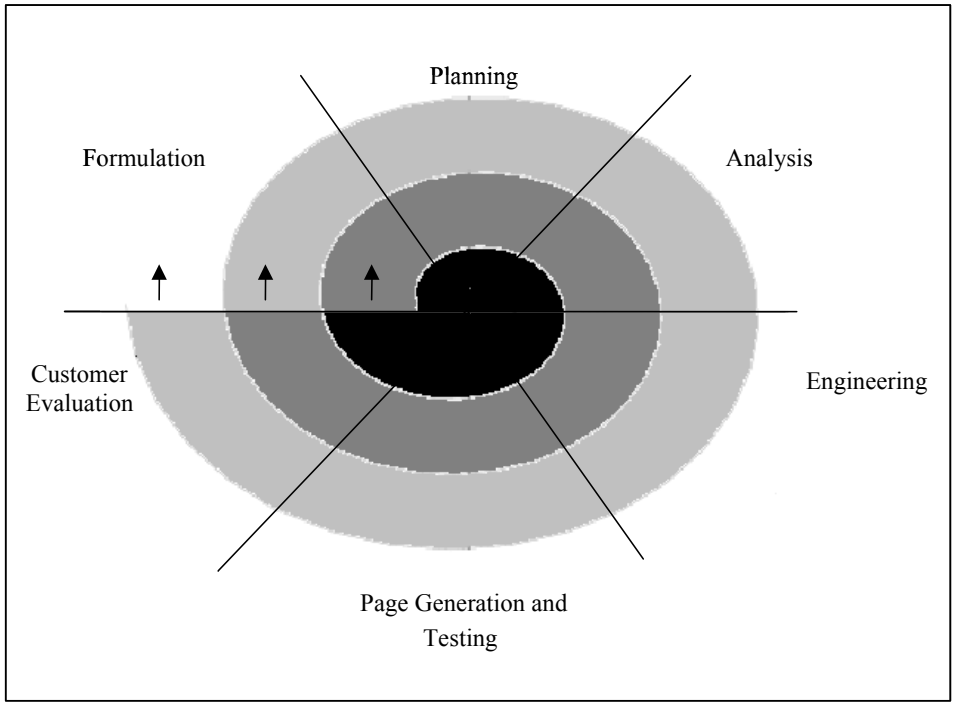

Figure 25. Spiral Process Model for Web-Based Applications

From this perspective, the Web-based application process begins with a formulation—an activity that identifies the goals and objectives of the application and establishes the scope for the first increment. Analysis is a technical activity that identifies the data, functional and behavioral requirements for the application.

The design phase has two parts as the engineering and the page generation. Engineering activity intends to design, to produce, and to acquire all data that are become

integrated in to the application. Page generation refers to a construction activity which combines all the data (content) and the necessary architectural, navigation, and interface designs. The middle-tier integration/implementation is also accomplished in this phase.

The customer evaluation takes place at the end of the each increment where feedback is received and changes are requested.

In Chapter II (Planning and Design), the motivation for the application and the initial requirements were defined. This was aimed to be the project's formulation, planning and analysis phase for the first increment. Later with the engineering and the page generation phase, the application was designed and a prototype was implemented. This chapter explains this phase of the thesis.

With the initial feedbacks from the customer (users), the other increment phases will continue.

A. DESIGN

The implementation was based on the initial requirements of the project defined in Chapter 2. The following assumptions were made for the simplicity of the initial prototype.

- The users were believed to have computer competency ranging from novice to expert.
- The user would have sufficient skills on using any Web browser.
- As the specification of the database was not concrete, the database and its relational tables could be designed and modified according to the future needs. And the implementation was built DBMS independent so that the system could swap the back-end DBMS without requiring extensive modifications.

1. System Architecture

The Educational Administrative Management System (EAMS) prototype was built on a three-tier client/server architecture. The benefits of this choice were discussed

in Chapter 3. The client interface was simply a Web browser where the user connects to the system and retrieves the data. In this architecture, all the business logic was implemented on the middle tier.

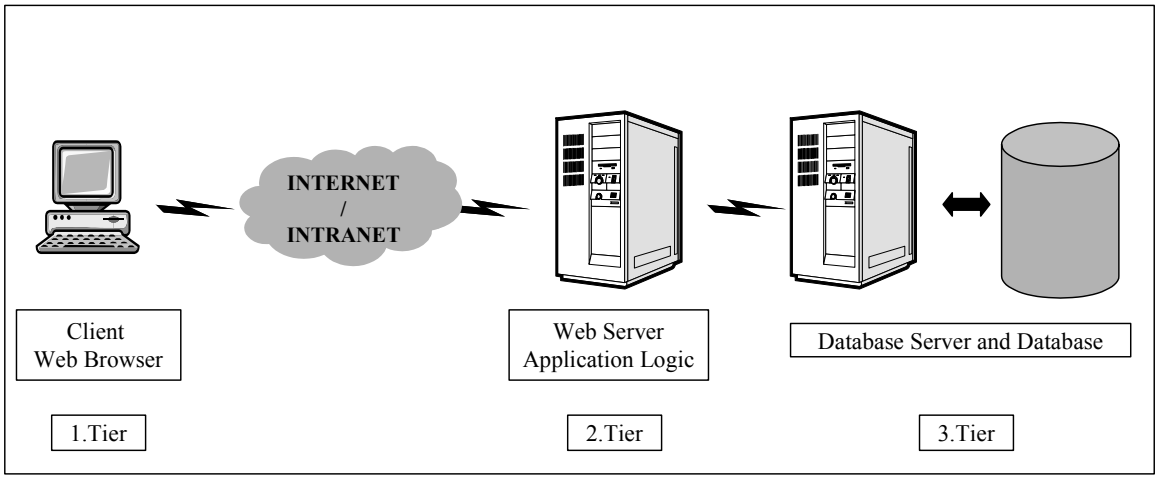

Figure 26. EAMS System Architecture

While Apache/Tomcat 4.1 was configured as a Web server, for the middle tier application, Java Servlets and Java Server Pages were used. Hall (2001) defines the Java Servlets as very powerful tools for the application logic, and the JSPs as very helpful for the presentation logic. Based on these properties of Java servlets and JSPs, the whole application logic was implemented in the Java Servlet where the presentation logic was divided into the related JSPs. The connection between the middle tier and the database was established via Java Database Connectivity (JDBC). A relational database instance was created on Oracle 8i Database, and populated for the implementation purposes.

2. EAMS Object Model

According to the use cases defined in Chapter 2, the EAMS' Object Model was developed. The basic objects in the system are defined and the interaction between the other objects determined. Figure 27 shows the simplified object model of the system.

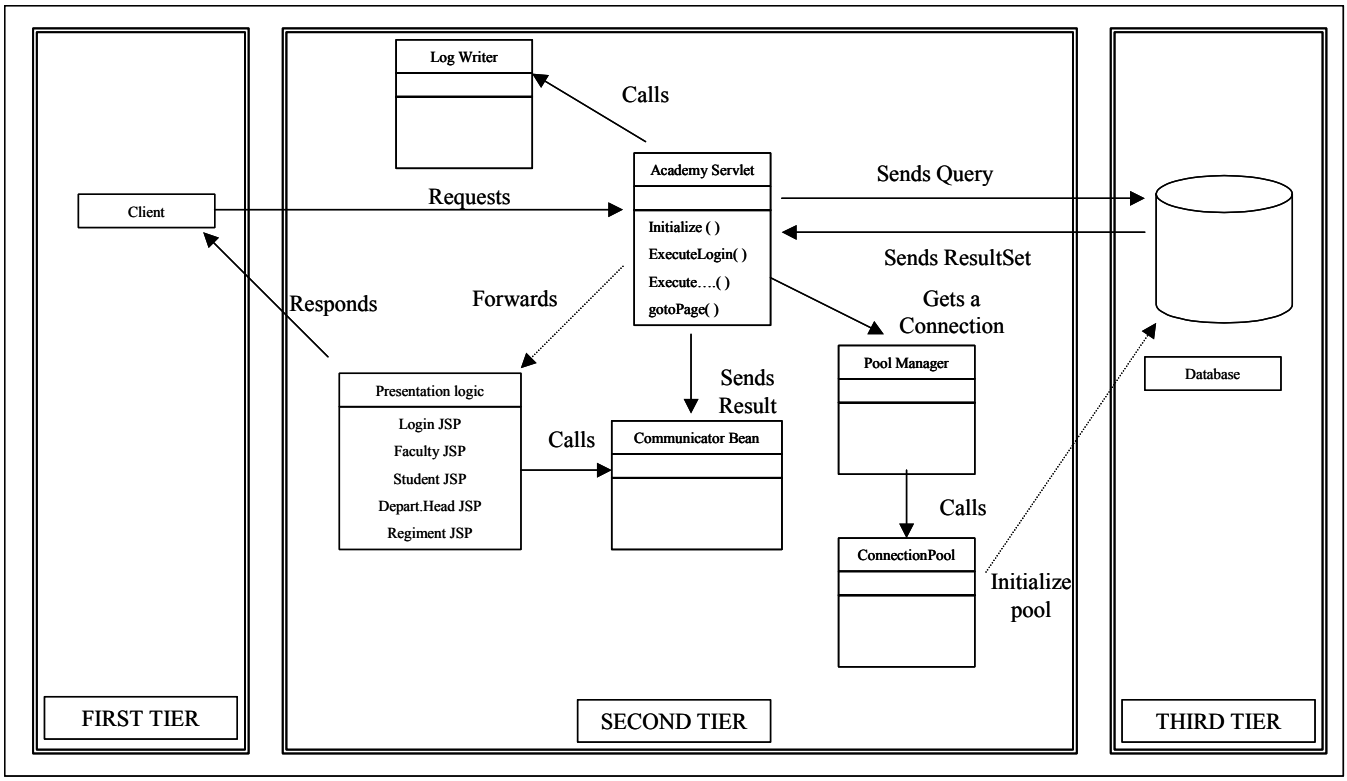

Figure 27. EAMS Object Model

The detailed explanation of the objects and their responsibilities will be discussed in later sections.

3. EAMS Database Structure

The first step for designing a database model is to discover the candidate entities. For this reason, based on the use cases in Chapter 2, the main entities of the database were selected. According to these entities, the relations between them developed. The designed database structure has four logical subdivisions. These are academic, regiment, personnel and system. Each subdivision has its unique tables and relations.

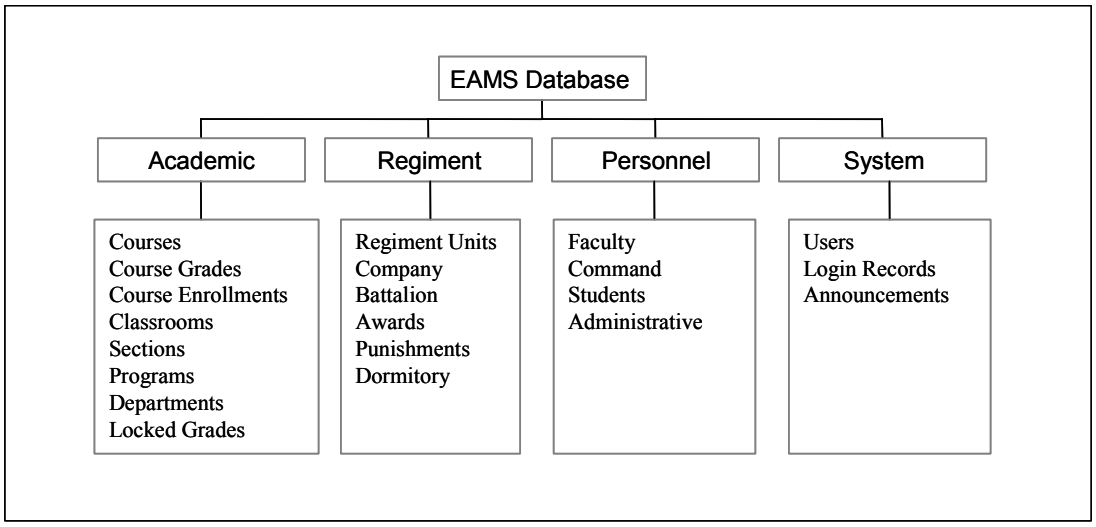

Figure 28. EAMS Database Structure

B. IMPLEMENTATION OF THE PROTOTYPE

In this section, the implementation of each tier will be discussed. As the client part is just a simple Web browser, implementation of the database was regarded as the first step during the implementation part.

1. Implementation of Prototype Database

The database for the EAMS prototype consisted of a database instance on an Oracle 8i database server. The structure of the database instance was explained in the design phase. According to the main entities, the related data subjects, tables and relations were created. The data subjects of EAMS database are listed in the following table.

EAMS Database Data Subjects		
No	**Name**	**Definition**
1	ANNOUNCE_CAT	Each user has different announcements categories. These categories are recorded in this table.
2	ANNOUNCEMENTS	Announcements are recorded in this table.
3	AWARDS	The awards of students are recorded in this table.
4	CLASS_ENROLLMENT	Courses, related sections, faculty members and the associated classrooms and as well as the scheduled dates are recorded in this table.
5	CLASSROOMS	Classroom numbers and their associated places/buildings in the campus recorded in this table.
6	COMMAND_PERSONAL	The information about the personal related with the regiment is recorded in this table.
7	COURSE_GRADES	The course grades of the students are recorded in this table.
8	COURSES	The detailed information about the courses are recorded in this table.
9	DEPARTMENTS	The information about the departments are recorded in this table.
10	DORMITORY	The dormitory room numbers and their associated places/buildings are recorded in this table.
11	FACULTY	The information about the faculty personal is recorded in this table.
12	LOCKED_GRADES	The dates of locking grades are recorded in this table.
13	LOGIN_RECORDS	Each new login and its time is recorded in this table.
14	PROGRAMS	The information about the programs in the academic department is recoded in this table.
15	PUNISHMENTS	The punishments of the students and associated information is recorded in this table.
16	REGIMENT	The regiment units in the academy are recorded in this table.
17	ROOMS	The rooms and their location are recoded in this table.
18	SECTIONS	The sections and their associated regiment units are recorded in this table.
19	STUDENTS	The information about the student is recorded in this table.
20	USERS	The users login Ids and their passwords are stored in this table.

Table 9. List of Data Subjects of EAMS Database

Once the tables and relations were formulated, the database instance was created by using SQL DDL (Data Definition Language). After creating the tables and relations, the instance populated with data for implementation purposes. To facilitate the understanding and documenting of the database, the relationship diagrams were also created. The relationships diagrams and the SQL DDL code for the database are provided in Appendix C.

2. Implementation of EAMS Application Prototype

The main purpose of this application is to support the Naval Academy's informational, educational and administrative tasks and their requirements conveniently and consistently. To accomplish this task, legal users will be given access to the system and according to their privileges and responsibilities, related menu options will be provided to them.

Based on this initial concept, by using a login page, the user logons the system. The login ID and password information are sent to the Servlet. The Servlet verifies this information with the database and obtains the user privilege code from the database. According to this privilege, the Servlet forwards a JSP file, which has the specific menu options for that type of user. Once users have accessed to this JSP page, by using related menu commands, they will access various operations in the system.

To develop this system, the Java programming language, Java Servlets and Java Server Pages APIs were used. JDBC and SQL commands were applied to the system for database communication. While Apache/Tomcat 4.1 was configured as a Web server, a database instance was created in Oracle 8i DBMS.

To have a better understanding of the application program, the components of the middle tier and the system operation scenario with the snapshots of the application will be discussed in the next section. The source code of the implementation is provided in Appendix D.

C. COMPONENTS OF THE MIDDLE TIER

The middle tier contains the following components.

- Http Server
- Login Page
- Academy Servlet
- ConnectionPool and PoolManager Classes
- Communicator Java Bean Class
- JSP Files
- Logger Class
- Database Properties File

The detailed information about each component will be given in the following section.

1. HTTP Server

As mentioned in the previous sections, the Apache Tomcat 4.1 Web server was used for the HTTP server purposes. The Web server enables the clients to download the HTML and Java Server Pages. This Web server can be downloaded from www.apache.org. Tomcat was sufficient for the EAMS prototype, but for the actual implementation, the performance, security, scalability and other related issues must be considered before selecting the HTTP server.

2. Login Page

To access the system, this is the first entry point for the user. This page was a simple input form asking for the login Id and password. For the implementation, a special JSP file was written, on which special login messages were shown to the user without invoking any extra application logic.

This page can be linked to any site that the user can call. Once the information is verified by the system, a new page will be displayed to the user. This page has the menu options for the specific functions of the system. It is important that, according to user's responsibilities and privileges, the subsequent page is different from the other users' pages. For example, if the user is a faculty member, after the login, he will reach a totally different page than students, regiment personnel, or department heads.

3. Academy Servlet

This is the main servlet that contains most of the application logic. All the user requests are addressed to this servlet where each one is forwarded to the appropriate method. After executing the appropriate function, the Servlet sends the results to the Communicator Java Bean, and forward a JSP file to the client. Later the JSP file gets the necessary information from the Communicator Java Bean, and shows the contents to the user.

Some of the main functions of the servlet can be summarized as follows:

a. Initialization

With the first call of the servlet, the servlets initializes itself.

b. Creating Database Connections

Even though a special class (Connection Pool) is used to create connections, the servlet creates and initializes this class. More details about the connections and the Connection Pool class are discussed later.

c. Getting the User's Parameters

When a user clicks on a menu option or a submit button, the command and the data are received by the Web server and sent to the Servlet. The servlet gets the information as a HTTP request parameter.

One of the important parameter that was used throughout the servlet is the "action" parameter. This parameter includes the information about what kind of a function the user requested. Some of the actions are login, get_Student_Grades and others.

d. Session Tracking

HTTP is a "stateless protocol". Each time a client retrieves a Web page, it opens a separate connection to the Web server, but the server does not automatically maintain contextual information about a client (Hall, 2001). This means that the Web server cannot easily remember previous transactions that the same user sent. This problem makes some applications like online shopping systems difficult. How can the system check and modify the user shopping chart?

In EAMS, session tracking has two meanings. Who is sending the request, and is the sender a legitimate user?

To solve the session tracking problem, cookies are used in EAMS. Whenever the user logs on the system, a cookie with some login information about the user is created. The same information is also recorded in the database. When a user sends a request to the servlet, the servlet gets this cookie from the user and verifies that the data matches the information in the database. If it is affirmative, the user request will be executed. If the user session does not contain the necessary criteria, the login page with an error is sent back to the user. The system wants the user to login one more time. Once the user logs off or closes the Web browser, the cookie terminates itself automatically.

More information on session tracking and cookies can be found in the reference (Hall, 2001).

e. Executing User Request

After the verification of the cookie, the request is sent to the appropriate method in the servlet. After the requested command is executed and the necessary data is taken from the database, the results are sent back to the user.

Sending the information to the user is not easy. The information must be presented to the user in a user-friendly way; it must be wrapped in an HTML page. For this reason, servlet sends back the information in two steps.

As a first step, the servlet loads the results into a Java Bean (Communicator). As a second step, the servlet checks the user type from the system, and then forwards the related JSP file to the user.

For example, assume that the user is a faculty member requesting the students grade information about a course. After the results are taken from the database, the servlet loads the result into the Communicator Java Bean. And since the user is a faculty member, the servlet forwards the Faculty JSP file to the user. The Faculty JSP file, containing all the presentation logic inside, gets the necessary data from the Communicator Java Bean and generates a HTML page. The user will see a simple HTML page with the information he or she requested with the other related commands in the Web browser.

More information about the Communicator Java Bean and JSP files will be given later.

4. ConnectionPool and PoolManager Classes

Creating a connection to a database is a highly time consuming since the database engine must allocate communication and memory resources as well as authenticate the user and establish a security context (Ayers, 2000). To solve this problem, systems share a set of already open connections between all clients.

For this purpose, two classes, the ConnectionPool and the PoolManager classes were created for the EAMS. While the ConnectionPool contains all the currently open

connections, the PoolManager interacts with the ConnectionPool and assign connections to the clients. The client (here a servlet) does not contact the ConnectionPool directly. When a user wants to run a query, the manager of the connections (PoolManager) will obtain a connection from the pool and give it to the user in order for the user to execute the query. When the query has been executed by the user, the connection is returned to the pool.

5. Communicator Java Bean Class

This Java Bean was used for communication between the servlet and the JSP files. When a requests is sent to the servlet, the servlets executes the request and prepares the related result. The result is later loaded in the Communicator Java Bean instance where the JSP file get the result and presents it to the user as an HTML page.

6. JSP Files

For complicated application that may require several substantially different presentations (like EAMS), a servlet can handle the initial request, process the data, set up Java Beans, then forward the results to one of a number of different JSP files.

In EAMS, after getting the results, the servlet forwards a JSP file to the user. All the presentation logic of the EAMS resides in these JSP files. The related JSP files receives the necessary information about the result from the Communicator Java Bean and creates a HTML page with it. The menu commands for each user is also kept in these JSP files and is shown to the user inside of the HTML page.

As there are four different user types in EAMS, there are four JSP files that contain the presentation logic for each one.

a. Faculty JSP File

The Faculty JSP file is used to present the faculty members' requests. This file has the following menu options:

- **My Home Page:** The initial page where the announcements related to the user will be shown can be reached via this option.
- **Courses:** The information about the courses that the user is currently giving will be shown via this option.

- **Personal Information:** The personal information of the Faculty member will be shown via this option
- **Class Schedule:** This option shows the class schedule.
- **Students:** This option shows the information about the students.
- **Submit Grades:** The user will use this option to submit grades to the students related to a course, which the user is currently giving.
- **Feedback/Problems?:** Any feedback or problems can be reported via this option.

b. Student JSP File

The Student JSP file is used to present the student requests. This file has the following menu options:

- **My Home Page:** The initial page where the announcements related to the user will be shown can be reached via this option.
- **Courses:** The option shows the courses that the user is currently taking.
- **Personal Information:** This option shows the personal information of the user.
- **Class Schedule:** This option shows the class schedule of the user.
- **Grades:** The user can see his grades by using this option.
- **Company Information:** This option shows the information about the regiment unit, discipline points, awards/punishments and the student's related commanders.
- **Feedback/Problems?:** Any feedback or problems can be reported via this option.

c. Department_Head JSP File

The Department_Head JSP file is used to present the department head requests. This file has the following menu options:

- **My Home Page:** The initial page where the announcements related to the user will be shown can be reached via this option.
- **Department:** This option shows the information about the department that the user is heading. The information about the instructors and the courses related with the department can also be reached via this option.
- **Personal Information:** This option shows the personal information of the user.
- **Students:** This option shows the information about the students in this department.
- **Instructors:** This option shows the information about the instructors in this department.
- **Feedback/Problems?:** Any feedback or problems can be reported via this option.

d. Regiment JSP File

The Regiment JSP file is used to present the Regiment command personnel requests. This file has the following menu options:

- **My Home Page:** The initial page where the announce-ments related to the user will be shown can be reached via this option.
- **Units:** This option shows the information about the units commanded by the user.
- **Personal Information:** This option shows the personal information of the user.
- **Students-Officers:** This option shows the information about the personnel (officers/students) commanded by the user.
- **Feedback/Problems?:** Any feedback or problems can be reported via this option.

7. Logger Class

For administrative and debugging purposes, a special logger class was used. This class provides the following benefits:

- Directing the logs to a file (in the server) we want to use
- Having more than one instance of a logger object for different purposes at a given time.
- Using time stamps for logging purposes
- Setting error levels or masking for the future reconfiguration of the system.
- Logging user information for administrative purposes.

8. Database Properties File

The database configuration parameters used for the ConnectionPool and the PoolManager classes are stored in this file. The name of the driver, the number of the maximum connections and the name of the log file can be stored in this file.

D. SCREEN SHOTS OF THE SYSTEM

In this section some of the screen shots of the system are presented. The similar functions and pages are not repeated for each type of user.

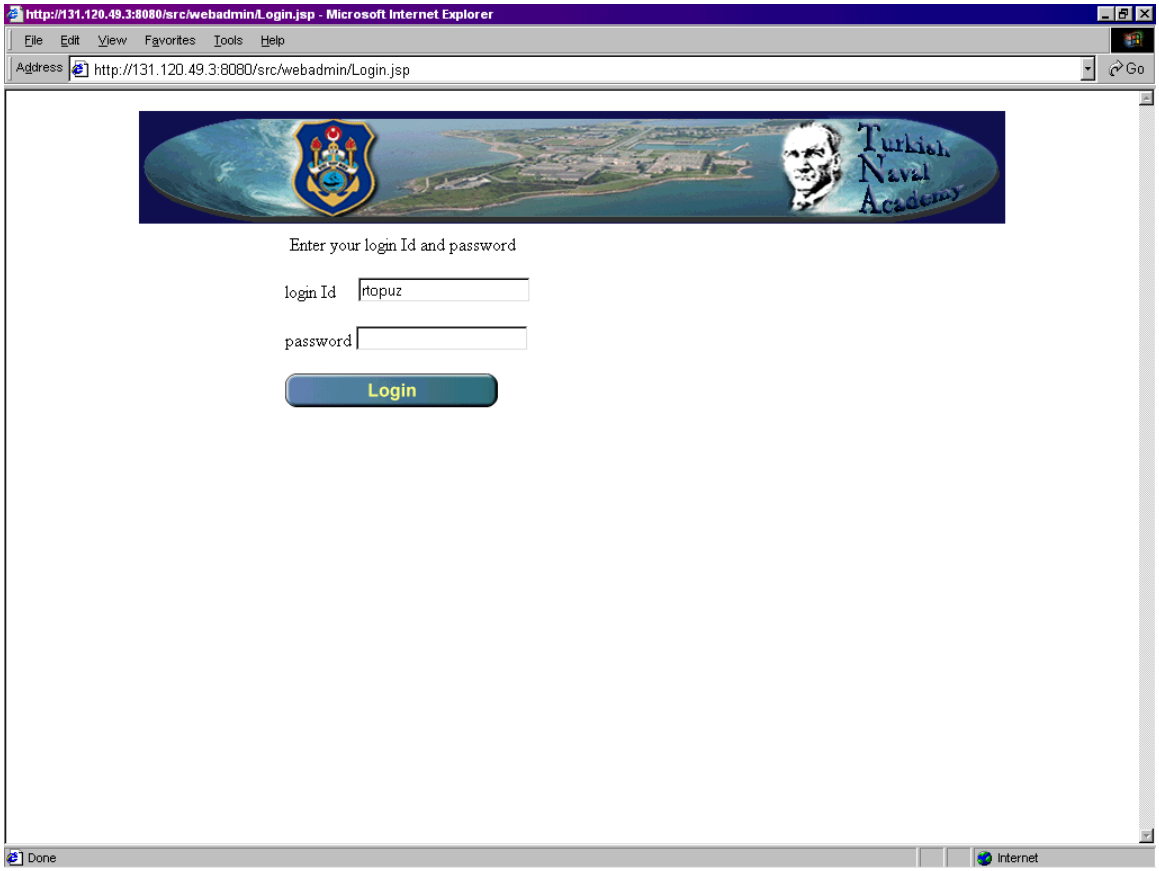

Figure 29. Screen Shot for Login Page

The first screen shot is for the login page, which is the only entry point for the system. All the users use the same login page. After entering the login Id and the password, the information is submitted to the servlet.

Figure 30. Screen Shot for My Home Page (for Students)

After the verification of the login information, the related JSP file is forwarded to the user. In this example, since the user was a student, the Student.jsp file was forwarded. The special menu options specific to the students can be seen at the left of the screen.

After accessing the system, the announcements (my home page) are shown to the user. The user can see this information anytime by pressing the “My Home Page” button.

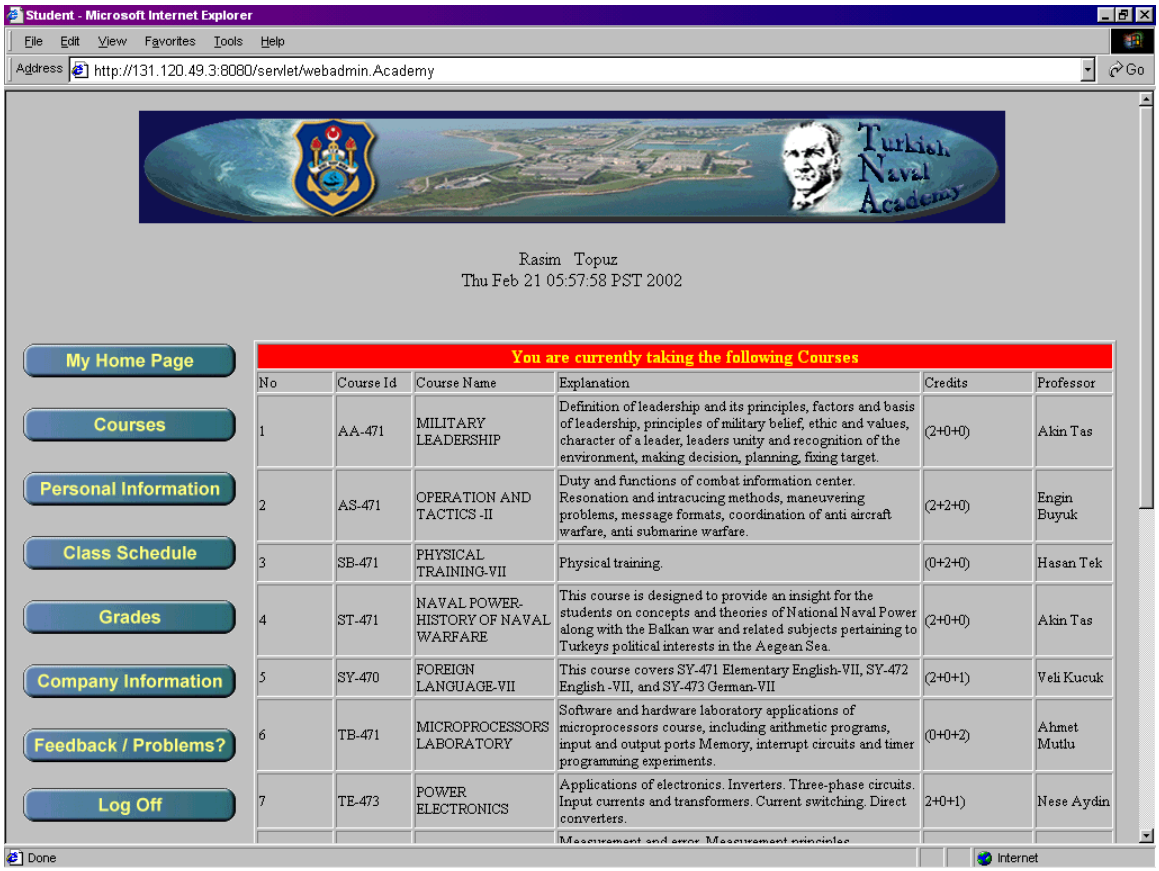

Figure 31. Screen Shot for Courses (for Students)

The users (student) can see the courses they are currently taking by using the "Courses" button in the menu. The courses are then shown on the screen.

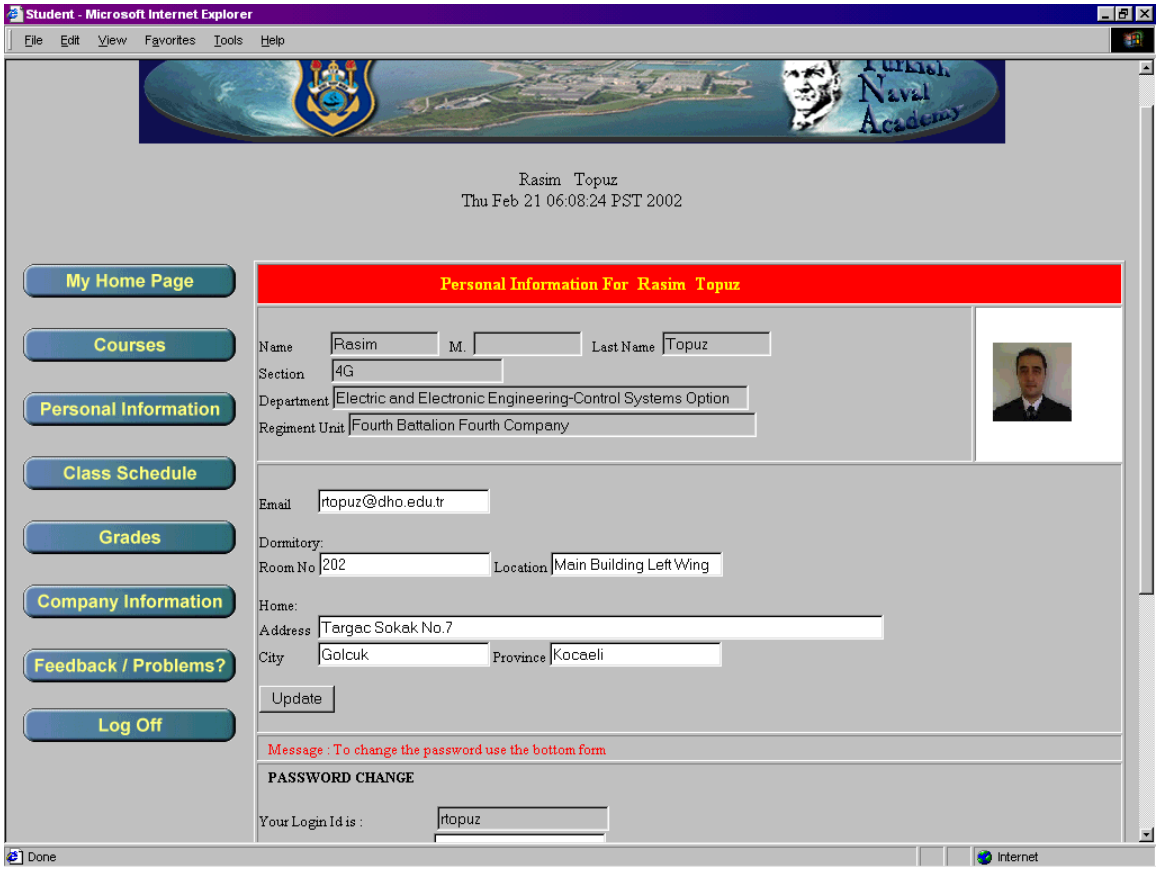

Figure 32. Screen Shot for Personal Information Page (for Students)

The users can see and update their personal information by using the "Personal Information" button. While some of the information cannot be updated, some other information can be updated by using this page. The users can also change their passwords by using this form.

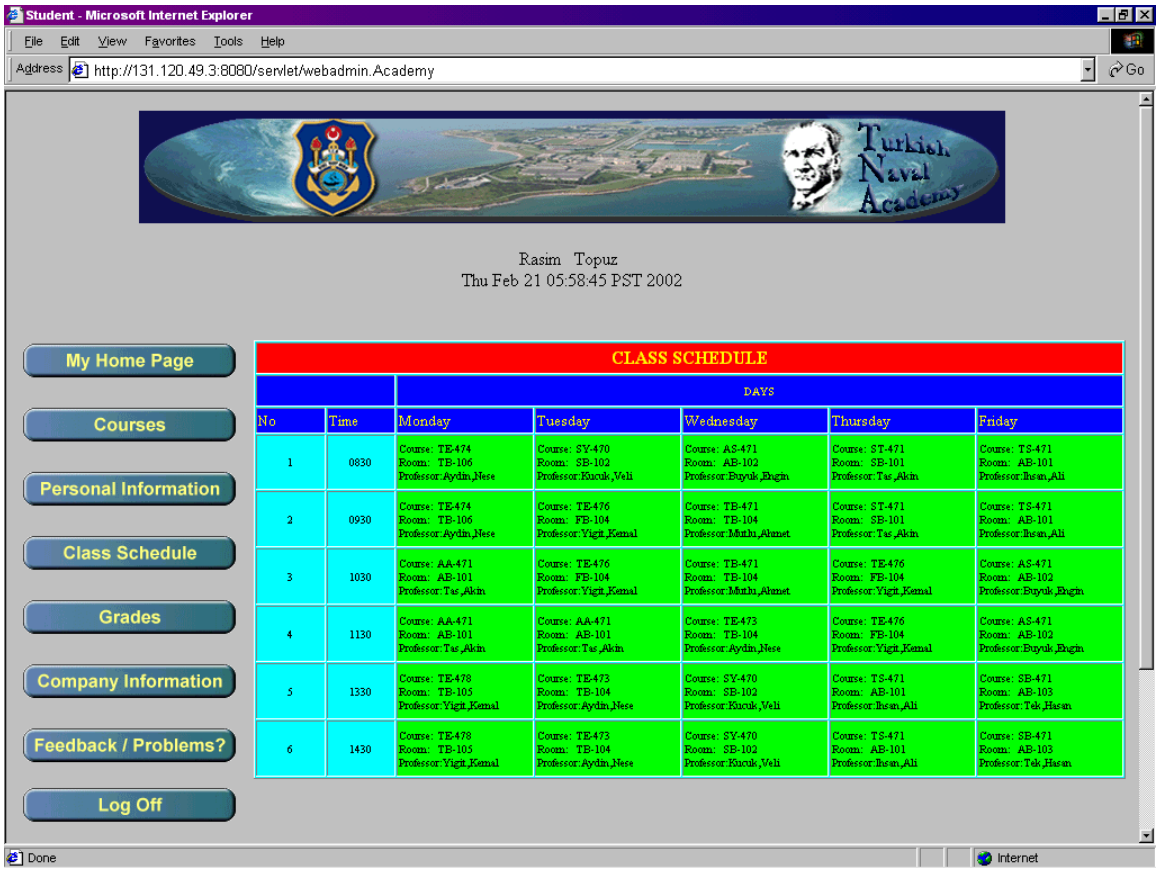

Figure 33. Screen Shot for Class Schedule Page (for Students)

The course schedule for the users can be accessed by using the "Class Schedule" button. After the data is retrieved from the database, information is presented in the actual weekly program by using a special algorithm. The users can see the course names, the instructors' name, and the classroom numbers.

Figure 34. Screen Shot for Grades Page (for Students)

If the users want to see their grades for any or all semesters, they use the "grades" button to see the following page. From default, the current semester grades and courses are shown. As an unofficial transcript, all grades can be seen via this option.

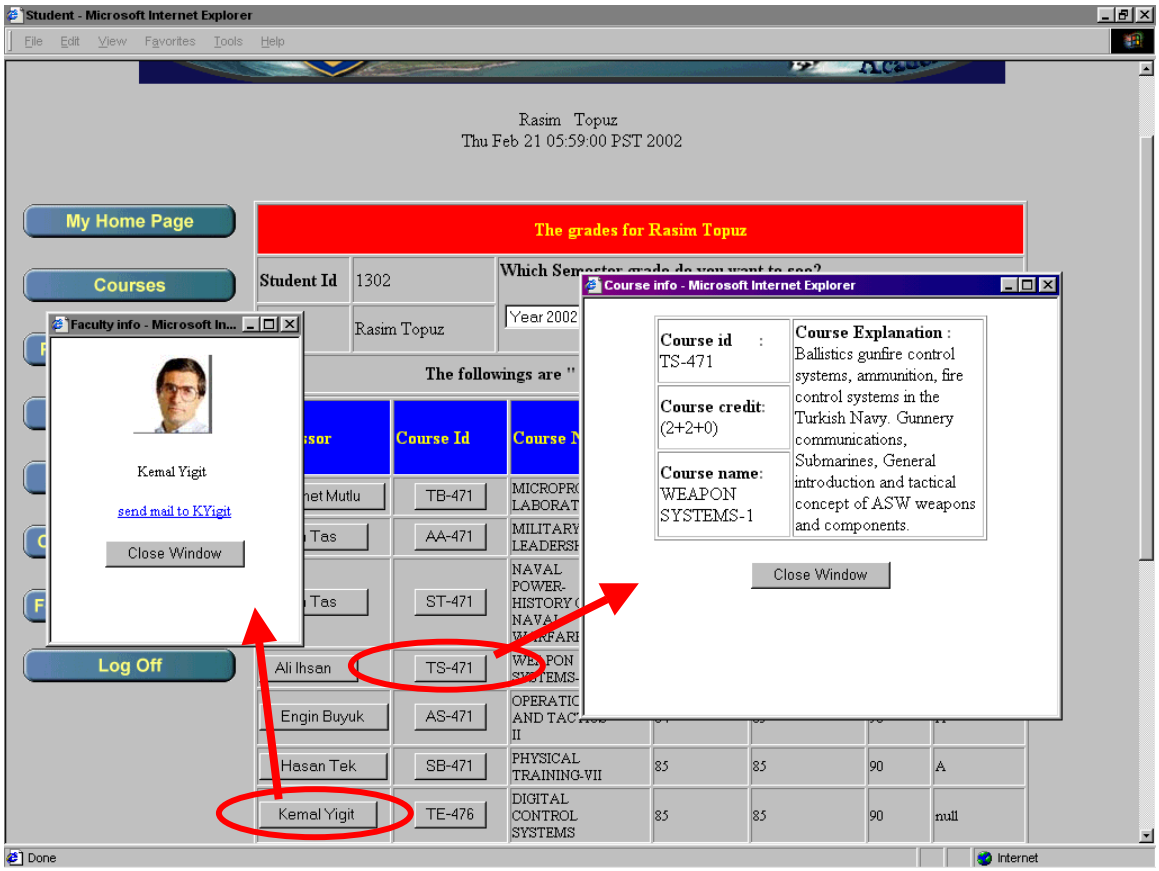

Figure 35. Screen Shot for Extra Information in Grades Page (for Students)

On the grades page, the users can also obtain more information about the instructor or the course they took by pressing the related buttons. The arrows in Figure 35 were used for explanation purposes of the screen shot. Pressing the related button will generate the output designated by the arrow.

Figure 36. Screen Shot for Company Information (for Students)

Since the scope of the academy's function entails military and academic training, data on both functions must be gathered and stored. Students belonging to various companies and battalions in the regiment can see their awards, punishments or remaining disciplinary points by using the "Company Information" button.

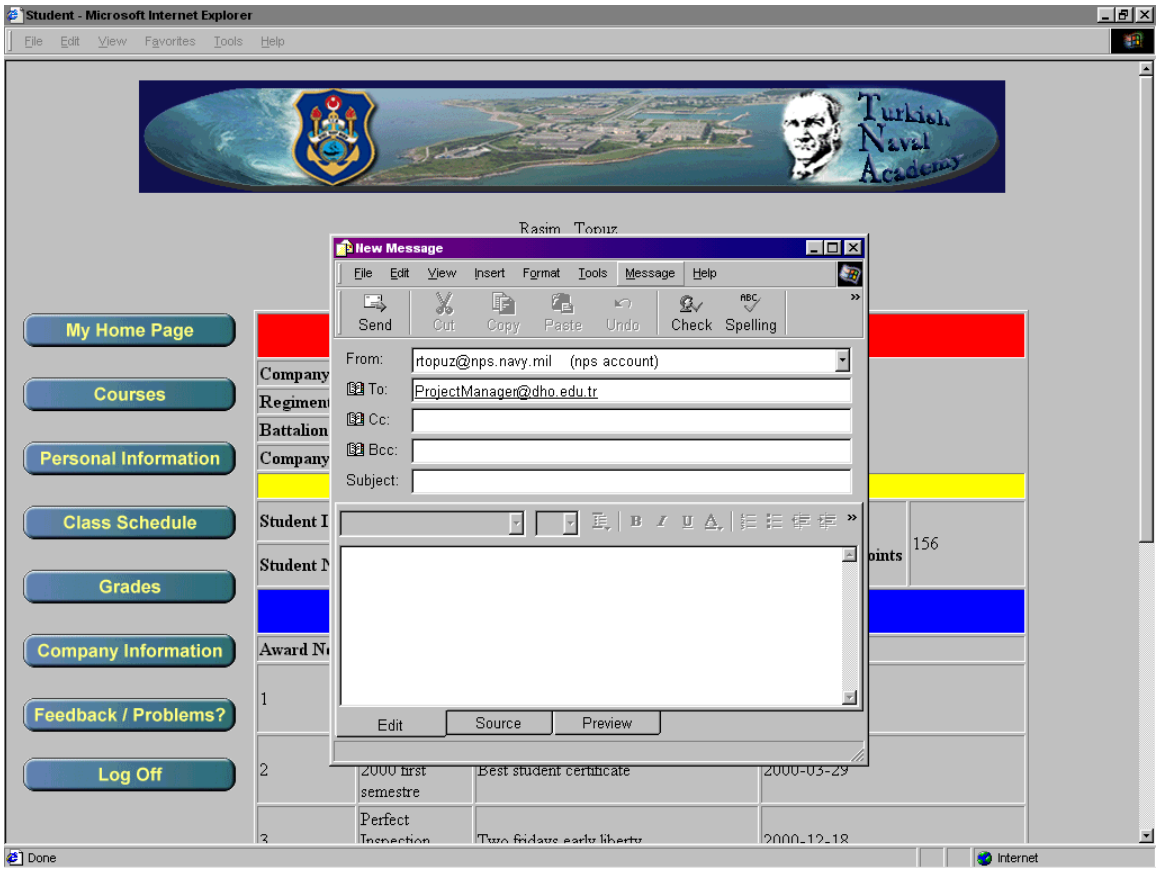

Figure 37. Screen Shot for Feedback/Problems Page

As the development process of the EAMS was defined as a spiral process, the users' feedback has greater importance. The feedback or problems can be reported to the project manager by using the related button. This button creates a new message instance where the user can easily report the problems or feedback.

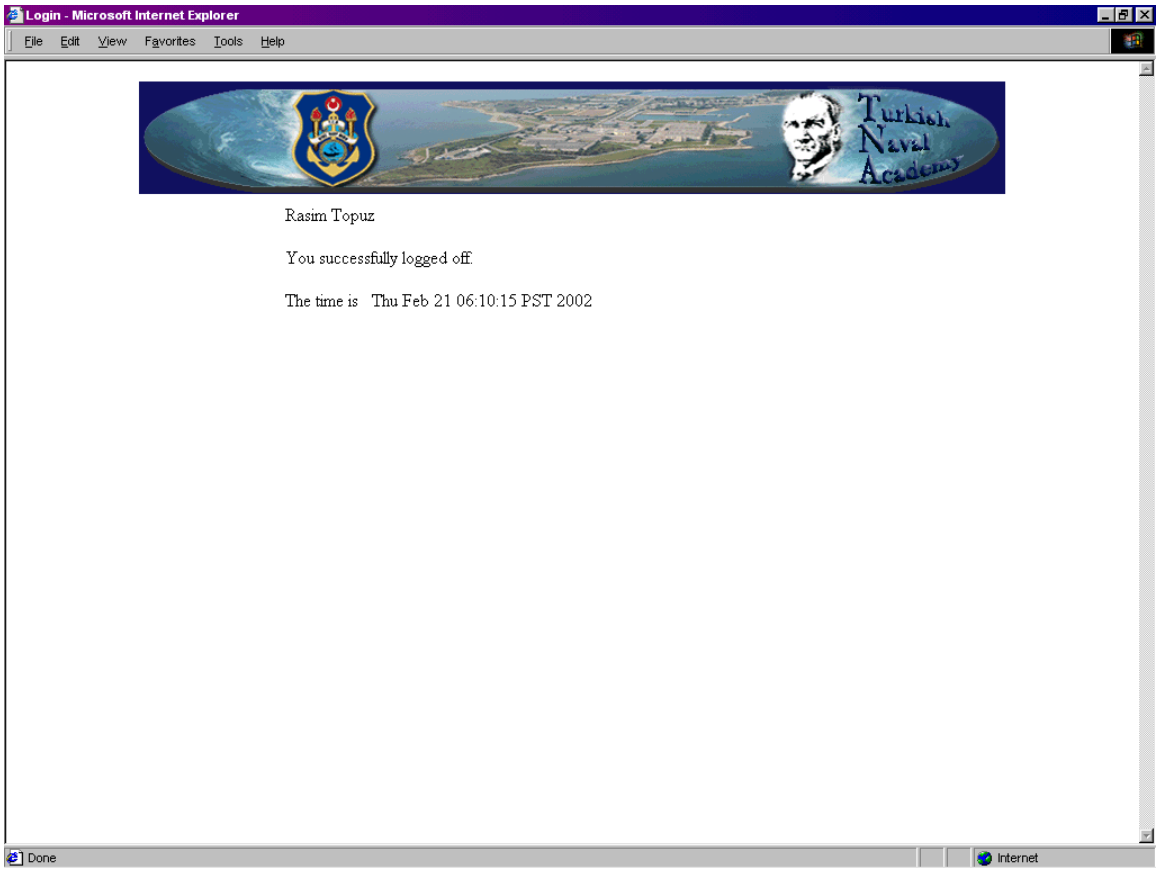

Figure 38. Screen Shot for Logged Off Page

The user can log off the system by pressing the "log off" button. A page will verify that the user logged off the system successfully. Without using this button, just closing the Web browser will also terminate the HTTP session and log off from the system.

Figure 39. Screen Shot for My Home Page (for Faculty Members)

Now a new user logged on the system. This user also used the same login page but since the user was a Faculty member, it can be easily understood that the menu options are different from the students' menu options. Again the first page is the My Home Page and the announcements.

Figure 40. Screen Shot for Courses Page (for Faculty Members)

By pressing the “Courses” button, the users can see the courses they are currently teaching.

Figure 41. Screen Shot for Class Schedule (for Faculty Members)

The Class Schedule button shows the users' schedule. Different colors distinguish between the empty slots and the filled time slots. The user can see the course number, the classroom number, and the section of the students taking the course.

Figure 42. Screen Shot for Students (for Faculty Members)

If the users want to see the students' information, they can see all students' grades by selecting the course name. As can be seen from the table, the user can sort the information by selecting any columns' button (such as, sort by name or midterm exam grade). In the screen shot, the information was sorted according to the course grade. If the users want to see the students' picture and email address, they can get this information, by pressing the related students' numbers.

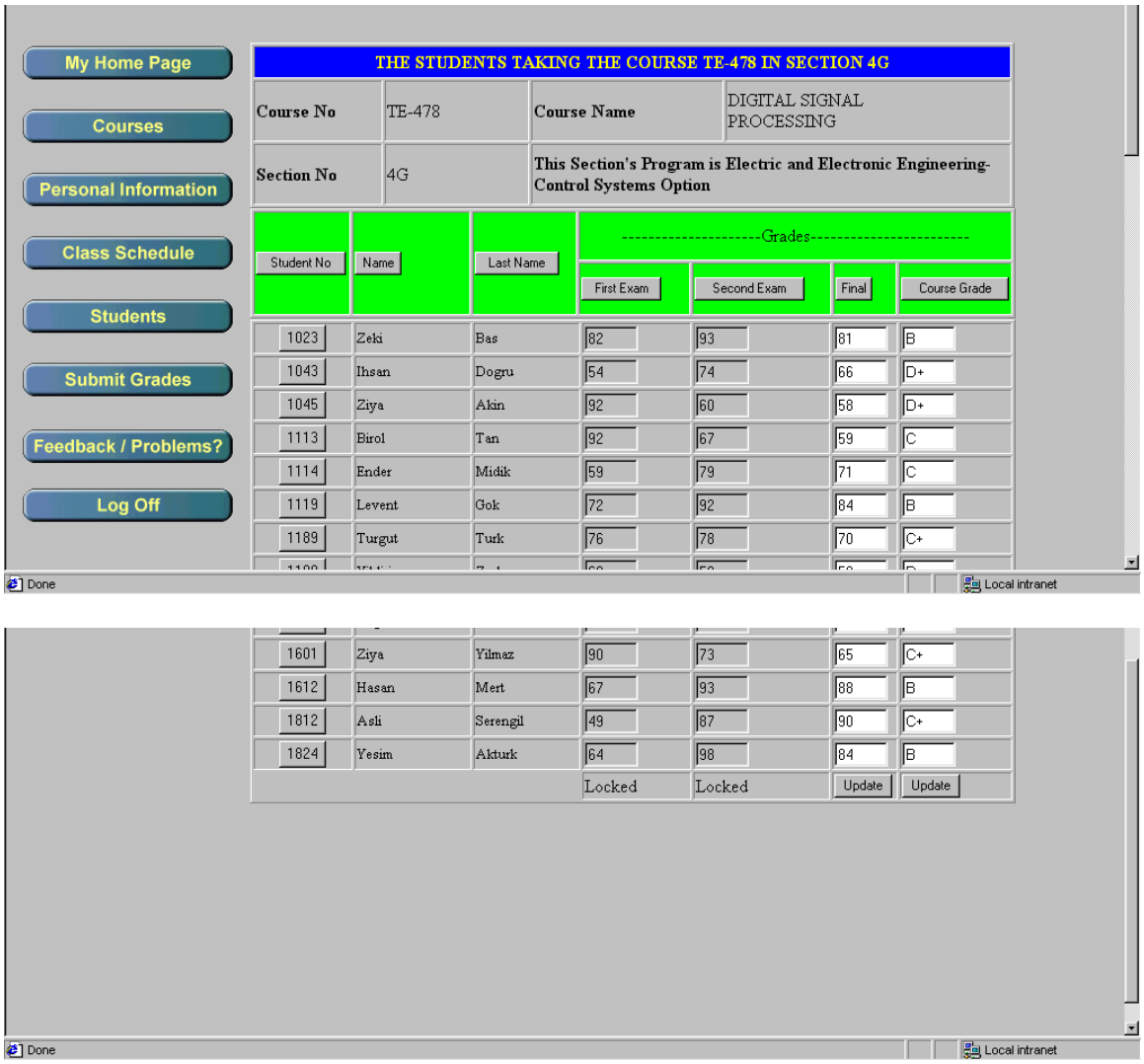

Figure 43. Screen Shot for Grade Update (for Faculty Members)

To submit grades for the students for a specific course, the user simply changes the grades for the student (or enters them for the first time) and then presses the update button to update the grades for related exam. As can be seen in Figure 43, some of the grades, here first and second midterms, cannot be updated because they are locked. This locking mechanism automatically activated because of a predefined date in the database for each exam to provide the consistency of the grades. After that date, those grades cannot be changed by using the application. A written request must be sent to the grade center.

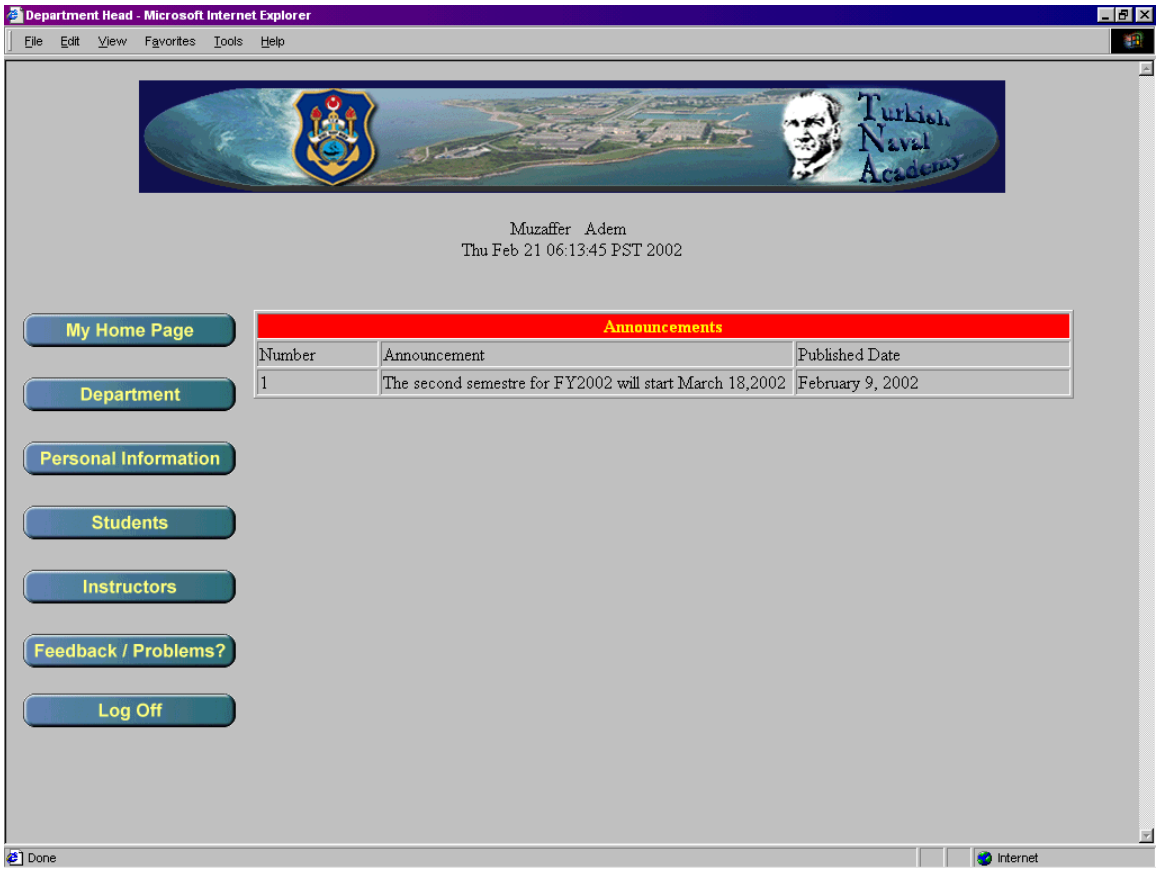

Figure 44. Screen Shot for My Home Page (for Department Heads)

The next user is a department head. The figure shows the My Home Page for the department Head. The unique menu options can be seen at the left of the screen.

Figure 45. Screen Shot for Department (for Department Heads)

By using the Department menu option, the user can see the department information and some basic information about the instructors, courses and sections related to that department. By pressing the info buttons, the user can receive extra information about the instructor, course or section.

Figure 46. Screen Shot for Students (for Department Heads)

By selecting one of the sections corresponding to a course, the user can see the list of all the students and their grades for that specific course. If the user wants to see the photo of the student or wants to send an email to the student, he can get this information, by pressing the student number.

Figure 47. Screen Shot for Instructors (for Department Heads)

Getting more information about the instructors is also very easy. By pressing the instructor's related button, the user can see the photo of the instructor or send him an email.

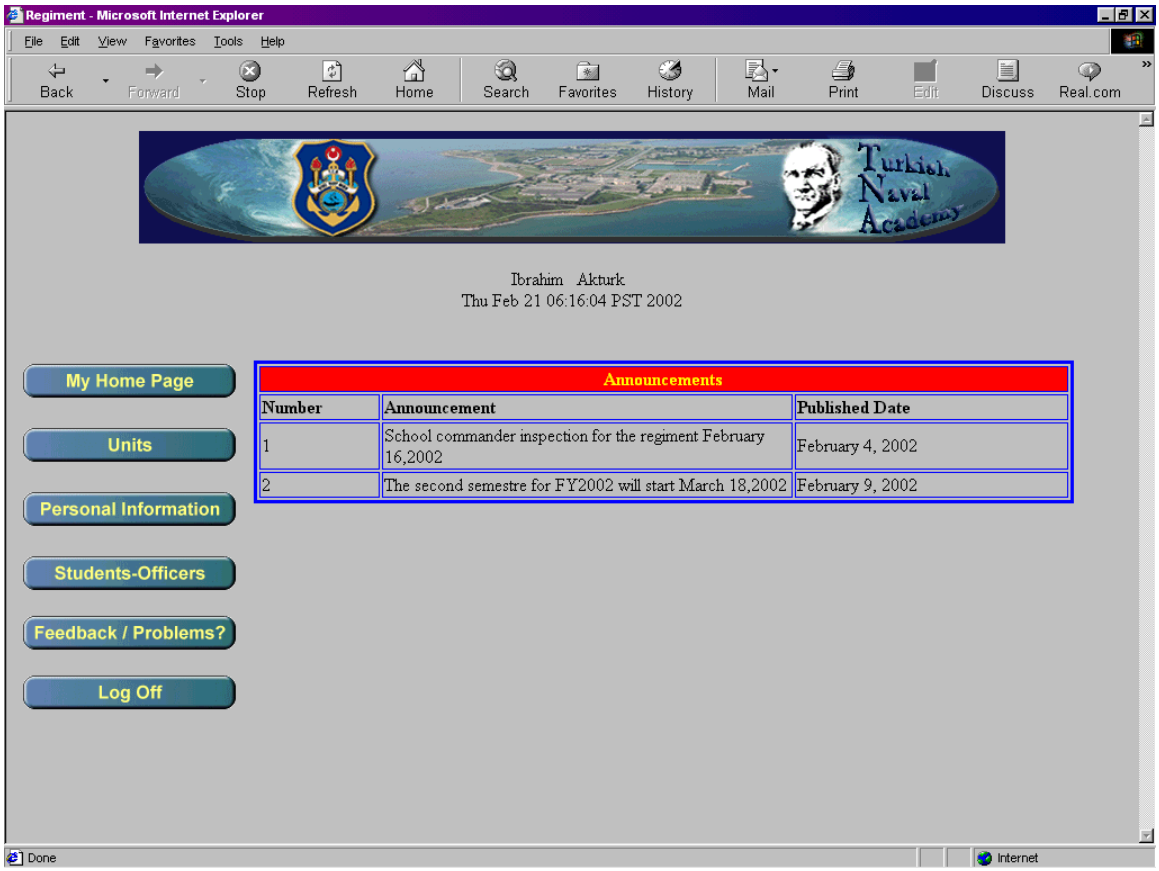

Figure 48. Screen Shot for My Home Page (for Regiment Personnel)

The next user type is a command personnel in the regiment. The home page is similar to the other, except for the menu options.

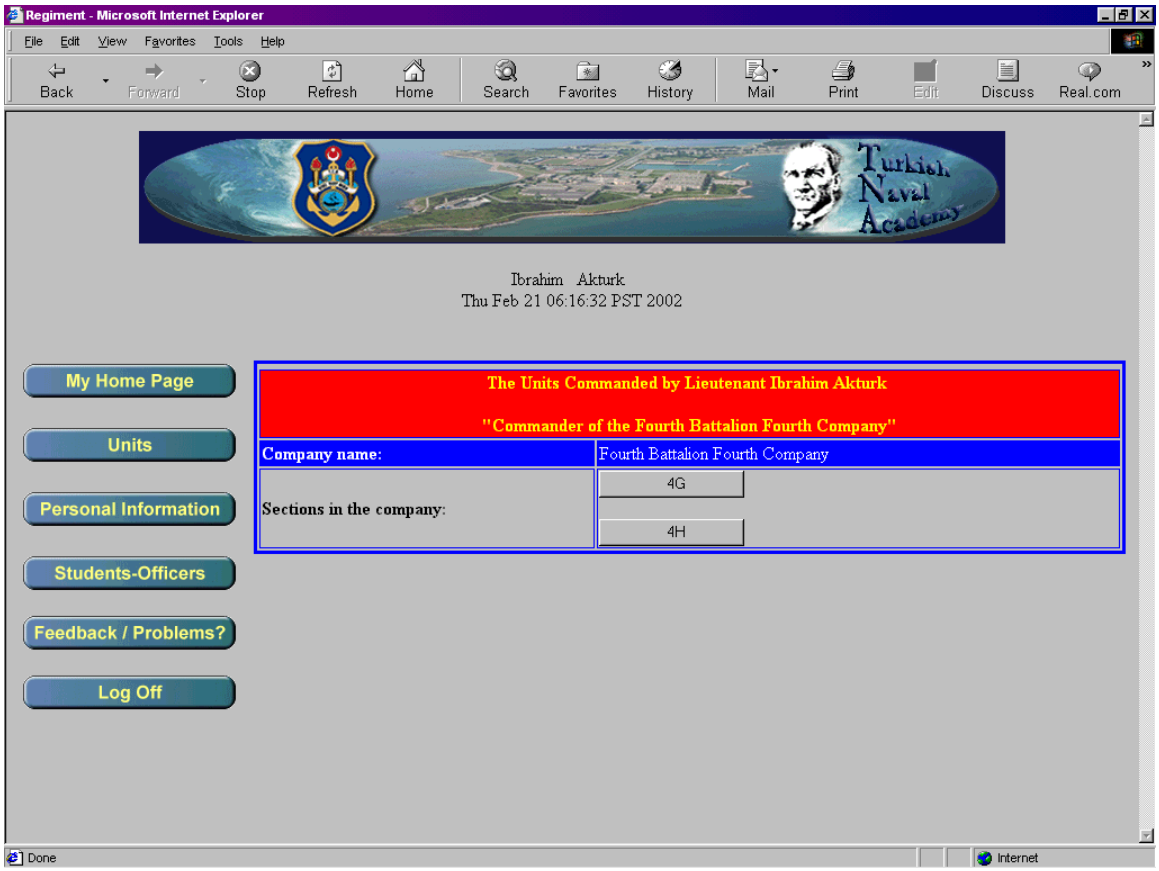

Figure 49. Screen Shot for Units-1 (for Regiment Personnel)

By using the units menu option, the users can see the units they command. As can be seen in Figure 49, the user was a company commander with only two units (in this case they are sections) under his command.

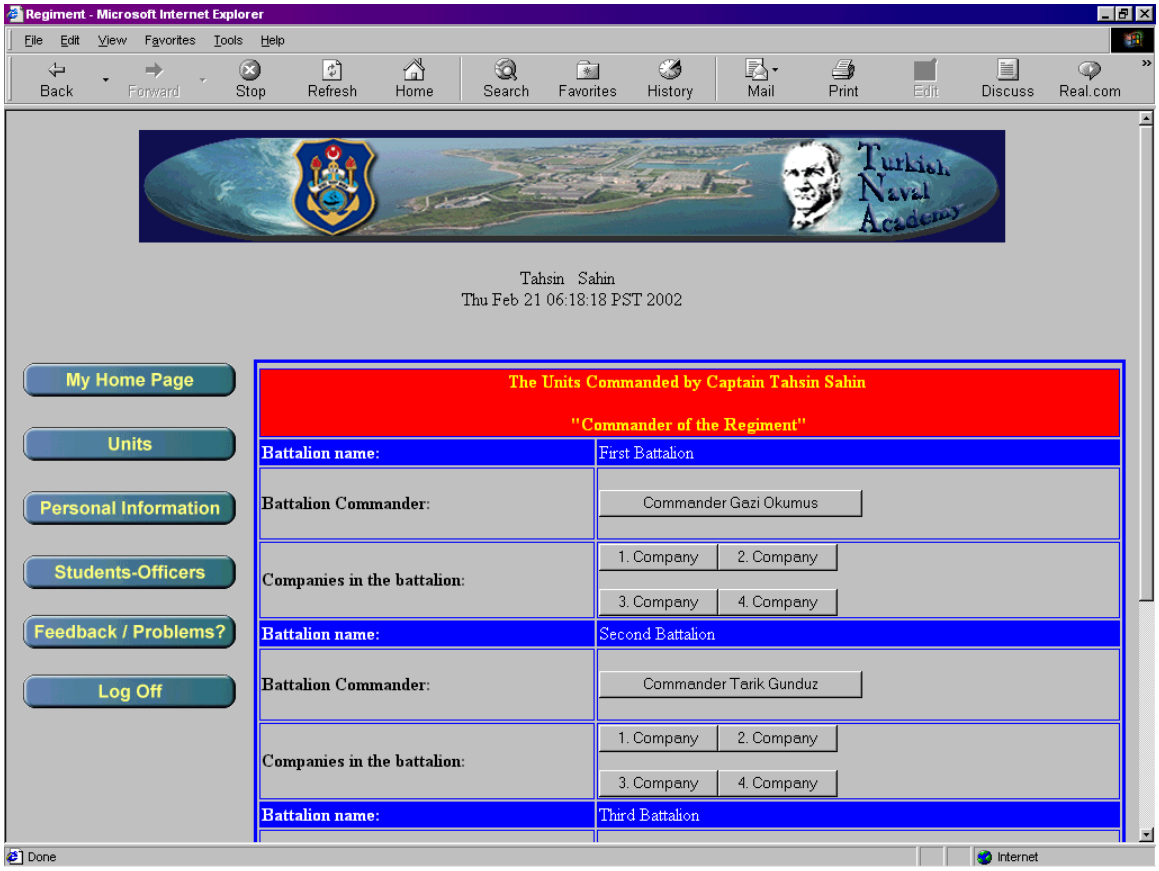

Figure 50. Screen Shot for Units-2 (for Regiment Personnel)

In this example another regiment personnel's units page is shown. This user is the regiment commander, who is commanding four battalions. The users can easily see information about any officer under their commands or by using the buttons for the battalion (or company), they can see the lower-level units and their personnel.

Figure 51. Screen Shot for Students-1 (for Regiment Personnel)

The user can see the list of all students in a section by simply selecting the company/section button on the units page. The figure shows the list of all students in Section 4G, which is in the 4th company. Under the chain of command, all related regiment personnel can access this information. In the next figure, we will see that by using the students' number button, more detailed information about the students can be accessed.

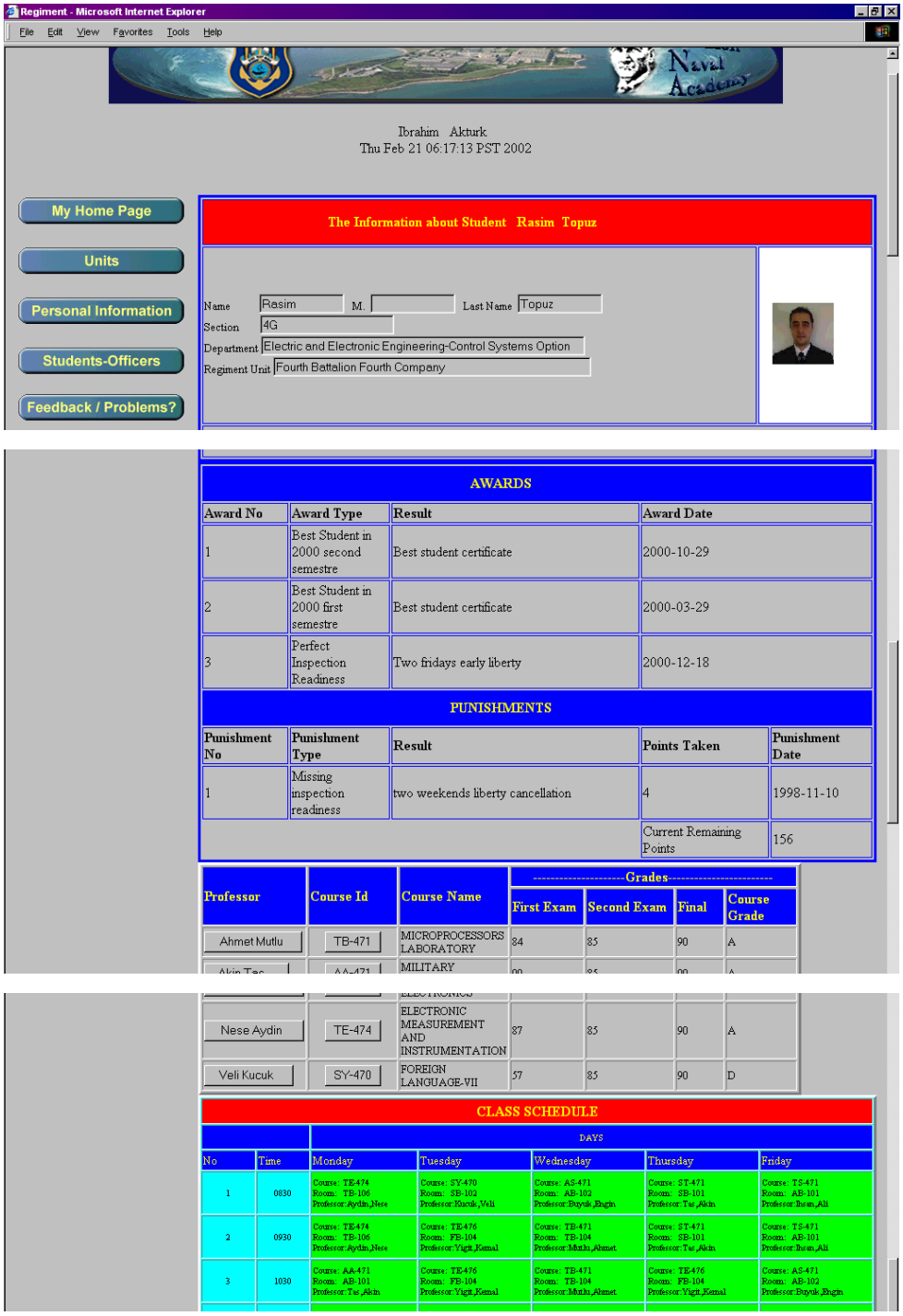

Figure 52. Screen Shot for Students-2 (for Regiment Personnel)

As shown in Figure 52, the user can see more detailed information about a selected student. All the students' academic, disciplinary, personal information can be accessed by using the related buttons.

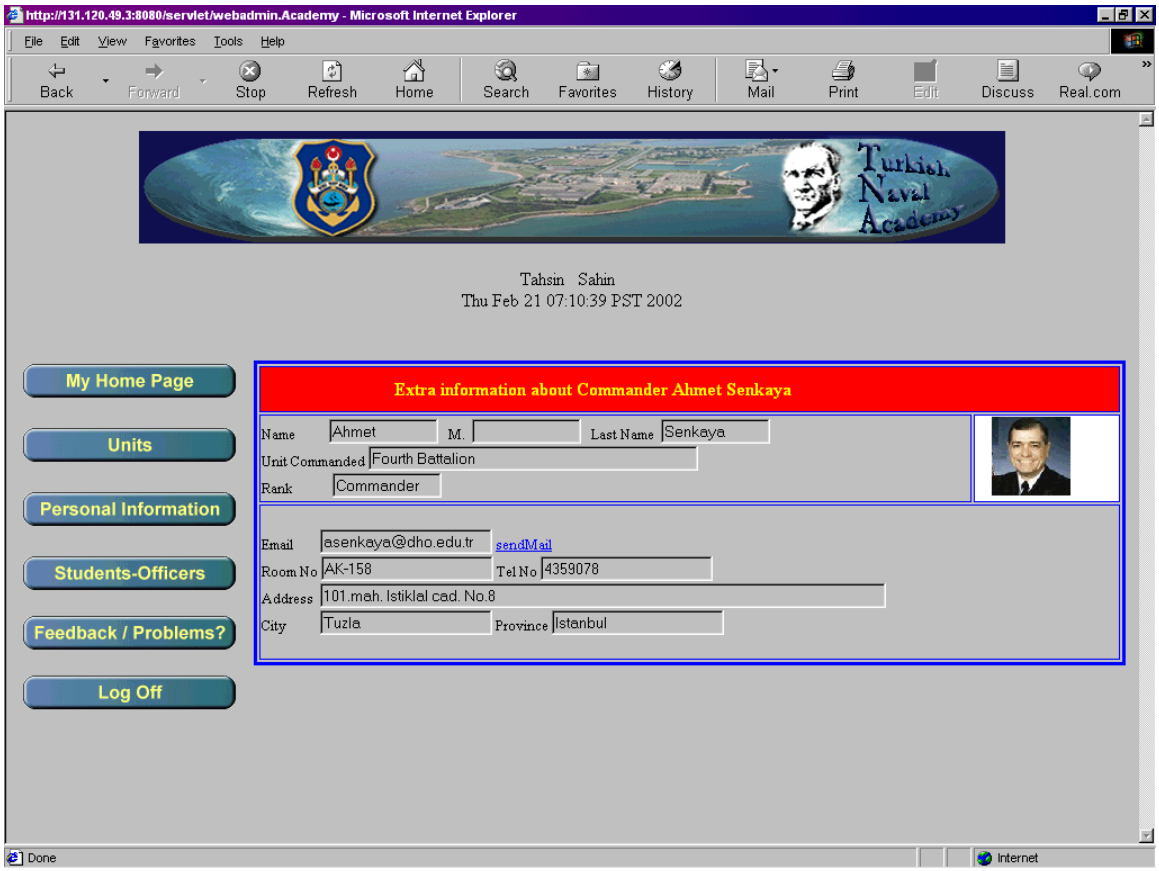

Figure 53. Screen Shot for Students-Officers (for Regiment Personnel)

If the user is commanding any officer and wants to see detailed information, this information can be accessed by using the officer button on the units' page. In this example, the regiment commander accesses information about the fourth battalion commander.

E. CONFIGURATION OF THE SYSTEM

Building a Web application is beyond merely writing code and running it. In order to configure the application successfully, the Web server must be successfully configured, both the application code and the images used in the pages must be located according to the configuration of the server.

For the configuration of the Web server, one must look at the Web server specifications. Since Apache Tomcat 4.1 was used for this application, the necessary configuration and the binary files for the Web server can be downloaded via www.apache.org.

For the implementation files, the JSP files have similar properties with the standard HTML files, so these files can be saved to any directory in the Web server. The servlets are Java files, so first they must be compiled and the binary code (class files) for the implementation files must be generated. And then these class files must be saved to the proper directory in the Web server. As Apache Tomcat automatically supports the servlets and Java, the class files for the servlets must be saved in the "WEB-INF/classes/" directory. It is important to note that only class files must be saved to this directory. It is better to save the source code somewhere else, not available from the outside. If any other Web server, other than Apache Tomcat, is used, it must be checked to determine if the server is supporting the servlets or not, otherwise, a servlet container application, such as JRun for windows must be used. In this case, the class files must be located at the appropriate location specified by the servlet container.

The images used in the system can be saved anywhere in the Web server, but in this case, the implementation files must address them properly. Otherwise the relative location of the files can be used in the implementation, and they must be recorded in this relative directory. In the EAMS prototype, the image files were assumed to be saved in the "/src/webadmin/" relative directory. So all the images can be stored in this relative directory (for the servlets and JSP files). In addition, since the servlets and JSP files will need the Java Virtual Machine, the proper location for the Java platform must be located in the CLASSPATH.

F. ASSESSING THE IMPACTS OF EAMS IN THE ACADEMY AND CHANGE MANAGEMENT

Even though the academy was using an online information system for a long time, enabling other users (students, faculty members and the regiment personnel) access to the system will change the way business has been conducted for many years. As this change involves people, there may be some reactions, even some resistance to the system despite, the support of the top-level management.

Mabey and Salaman (1995) consider a number of perceptions about the management of change that affect reactions to it. Among these factors are whether change is perceived as "deviant or normal" and "threatening or desirable." Perceptions about the nature of change and the need for it therefore affect reactions to it. The perceptions can be different, but in each case, the change management issues must be addressed during the process.

Beckhard (1992) defines ten organizational requisites for the change management process. These are

1. Ensuring senior management commitment to the proposed changes, which must be visible to all participants throughout the organization.

2. Producing a written statement about the future direction of the organization that clarifies its objectives and values throughout the organization.

3. Creating a shared awareness of conditions to produce a common perception that change must be implemented.

4. Assembling a body of key managers and other important opinion-formers to gain their commitment to the change process so that this may be disseminated more widely.

5. Generating an acceptance that this type of change will require much time to implement fully even though there may be short term, dramatic changes as part of the overall transformation.

6. Recognizing that resistance to change is part of the normal process of adaptation so the managers can be educated and made aware of this and equipped to manage this reaction.

7. Educating participants about the need for the change and training them to be competent and effective and to overcome resistance and make commitments.

8. Persevering with the change process and avoiding blame when an attempt to implement a facet of this process fails. Such negative action generates resistance and reduces the necessary risk-taking behavior.

9. Facilitating the change process with necessary resources.

10. Maintaining open communication about progress, mistakes, and subsequent learning.

It is believed that the eagerness and good will of the personnel, the military structure of the academy, and the support of the academy commander will make it easier to implement the EAMS in the Naval Academy.

Based on the previous requisites, the following steps should be followed during the implementation/configuration of EAMS in the academy:

1. Find a Champion

The first step in managing change is getting the support of a champion, (a top-level manager) who has the power and the authority to promote the project.

A champion must be the highest-level line executive or commander who is willing to put his or her moral capital at risk to provoke enthusiasm of an IT system. An IT manager cannot be a champion (Haga, 2002).

In the academy this champion is the academy commander. Without the commander's support and authorization, such a system cannot be implemented.

2. Emphasize the Need

In order to produce a common perception that the change must be implemented, the problems with the old system, and the potential benefits of the new system must be made clear to all users.

3. Form a Powerful Guiding Team

A team, including the key managers in each department, is also very important for the implementation. This team can organize and identify the healthy feedback and evaluations.

4. Recognize the Resistance and Deal with It

Recognizing the resistance as normal is another step in the change management. With this understanding, the middle managers, such as the department heads or regiment battalion commanders, can be informed and trained to manage this reaction.

5. Educate All Users

To overcome resistance, the users of the EAMS can be trained and educated for both the benefits and the efficient use of the system. This can be accomplished by a series of conferences where a demonstration for the system will be done for each user type.

6. Maintain Open Communications

The feedback from the users at any stage of the implementation has great importance. Not only usability issues but also any possible problems in the system can be identified. Also the user can be informed about the progress of the system or any future enhancements.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSION

A. SYNOPSIS

With the introduction of the Internet, the information systems faced major changes. Accessing the information anywhere and anytime is one of the key benefits of the Web-based applications. Web-based applications are proven to be scalable, high performing, and easy to implement, to manage and to update.

This thesis presented the design, development, implementation and analysis of the Educational Administrative Management System (EAMS) for the Turkish Naval Academy. EAMS will provide the academy with broader information availability, better performance, and a decrease in paperwork and manpower.

The thesis also covered the current situation and the need for such a system. The main goal of EAMS is to reduce the loss of productivity by increasing the availability, accuracy, efficiency, and the consistency of the information related with both the administrative and educational data in the naval academy. This system is believed to eliminate most of the current drawbacks and to result in savings for personal power and time while quickly providing the required information to the users.

Each step of the development cycle of the implementation was discussed in this thesis. Starting from the architecture evaluation, the middle-tier technologies, database connectivity tools and databases were studied.

According to the studies, a Web-based three-tier architecture was found as the best model for the system. As the client tier was a simple Web browser at a PC or a PDA (personal digital assistant), the main emphasis was given to the middle tier and the database tier.

From the comparison of the middle tier technologies, Java Servlets and Java Server Pages (JSP) were selected as the best combination for the system. While the application was designed in the servlet, all the presentation logic was defined in the JSPs.

As the spiral process model was used for the implementation, a prototype for the system was also implemented. According to the user feedbacks, future enhancements will be done in the next iterations.

For the database tier, a relational database model was selected. With the definitions of objects and the relational schema, an instance of the database for the academy was created in an Oracle 8i database. The database instance was also populated for implementation purposes.

Such a system can be implemented in various areas. From Naval Supply centers to personnel departments, such application can be used to reduce paperwork and manpower.

B. FUTURE ENHANCEMENTS

1. Adding More User Types to the System

For the implementation purposes and to get accurate information from the academy, the administrative office users system interfaces were not defined in the requirements. These users must have special features that will enable them to print the academy's official records and reports. According to the format of those reports and with the additional requirements, this user type can easily be added to the system.

Another user type for the future enhancement is the system administrator who can be given the capability to see the current online users, their activities and to perform any online support/maintenance.

2. Enhancing the Security Features

Currently, the system uses cookies for session tracking. In this method, the user login Id and the login time stamp is compared with the original system records. For the security purposes, this information is verified each time the user makes a request to the system. As an additional security feature, the access to the system can be limited to the intranet. This can be easily done by checking the IP address of the client in the implementation or by configuring the Web server accordingly.

Even though each user has different privileges, the same database connection pool is used in the prototype. As the system can resolve this privilege issue by forwarding the special JSP files to present the information, however creating different connection pools for each user type can enhance the security.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. FIVE STYLES OF CLIENT/SERVER COMPUTING

In this thesis, client-server models were discussed under two-tier and three-tier client server architectures, and then compared. It is important to note here that this representation (2-tier or 3-tier) is not the only representation for the client-server architectures.

Another model for client-server architecture is the Gartner Group Model of Client/Server Computing. (Goodyear, 2000) This model is based on dividing business processing into three distinct layers: database, application logic and presentation.

In the following figure, the solid line labeled "Network" shows which of the five layers are allocated to the client and which are allocated to the server. The upper part of the "Network" designates the server side; the lower part of the "Network" designates the client side.

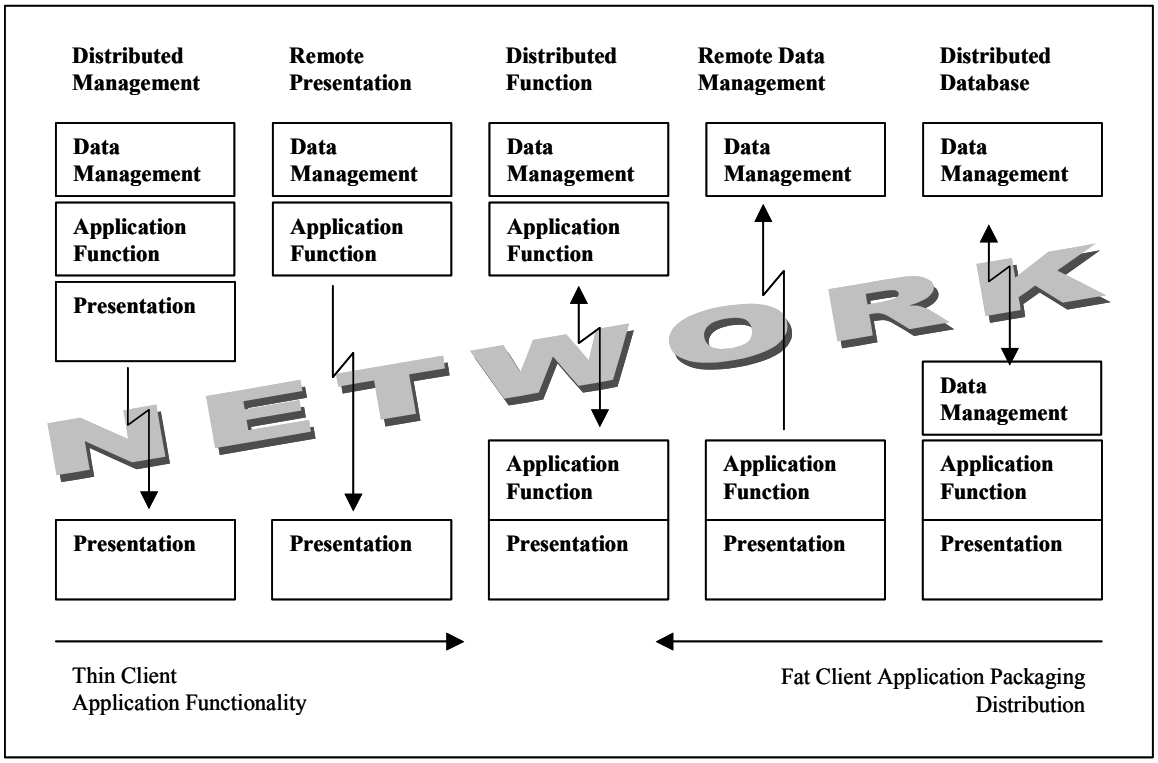

Figure 54. Gartner Group's Five Stages of Client/Server Computing (From Goodyear, 2000)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. WEB ARCHITECTURE

To open a Web page in our browser, we usually either type in the URL or click a link to the URL. What happens after this point? How does the browser connect to the server? How do we see the Web page in our browser?

This appendix was aimed to give the necessary background to answer these questions about the Web architecture and its components. Before getting involved with details, we should examine the simple request and respond flow between a browser and a server.

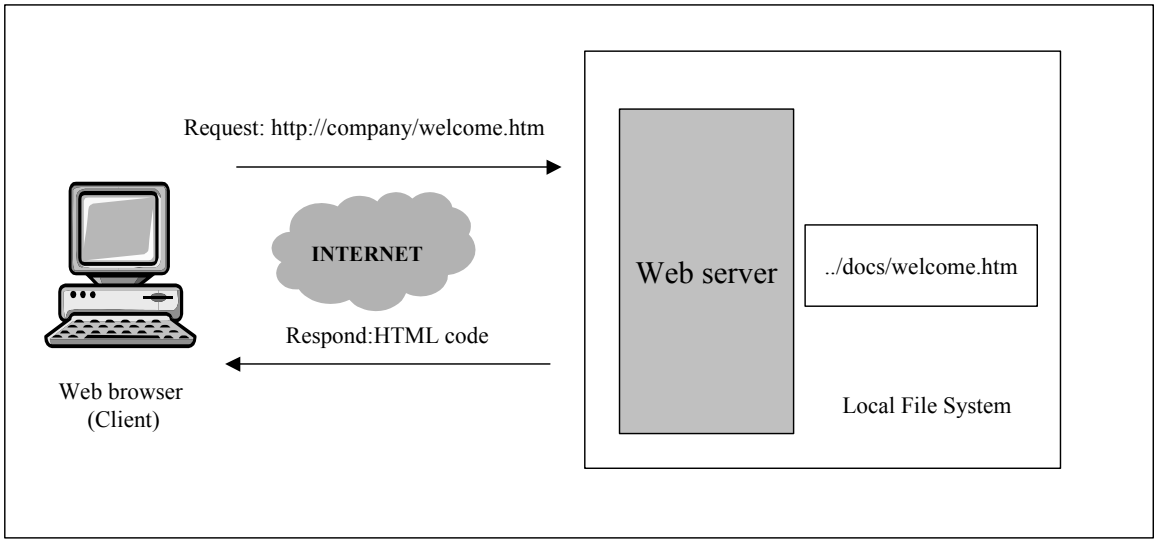

Figure 55. Simple Request and Respond Flow between a Browser and Web Server

In the Figure 55, our URL is http://company/welcome.htm. Once we submit this request and the Web server receives it, the Web server locates the Web page in its local file system and sends it back to the browser. The browser then displays the page. Each image in the page is also referenced by a unique URL, and the browser requests each image's URL from the server in the same way that it request the main html file.

At this point, we see that Web browser and the Web server are the main entities in the Web architecture. It is better to look at each of these entities.

1. Web Browser

A browser is a computer program that resides on our computer and enables us to use the computer to view World Wide Web (WWW) documents and to access the Internet, taking advantage of text formatting, hypertext links, images, sounds, motion and various other features. The Web browser can be thought of as a universal user interface. Whether we are doing some simple Web browsing or the transacting online banking, the Web browser's only responsibilities are that of presenting the Web content, issuing requests to the Web server, and handling any results generated by the request.

Browsers cannot only show us the HTML pages but can also show others such as Extensible Markup Language (XML), dynamic HTML (DHTML) pages. It is important to clarify that all these markup languages were based on SGML (Standard Generalized Markup Language-ISO8879), the international standard for defining descriptions of the structure and content of different type of electronic documents.

The Web browsers can also execute applications within the same context as the document on view. The two popular examples for client-side Web applications are Microsoft's ActiveX technology and Java applets (Ayers, 2000). ActiveX components are downloaded from the Web server, registered with the windows registry, and executed when called by other script elements. A Java applet is a small program also downloaded from the server and executed within the browser's own Java Virtual Machine (JVM). Both ActiveX objects and Java applets have full access to the browser's document object model and can exchange data between the browser and themselves. The main difference between them is the accessibility and security issues. ActiveX controls make it possible for a Web browser to interact with other desktop applications. Because of this, any possible security hole in the application can invite hackers to seize control of a computer system (Cert, 2000). Contrary to this, as the Java applets are running in JVM, their capabilities are restricted for security concerns.

Two Web browsers dominate the Web browser market. These are the Microsoft Internet Explorer and the Netscape Navigator.

2. Web Server

The Web server is a program running on the server that listens for incoming requests and responds to those requests as they come in. The web server is assigned an IP address and is connected to the Internet so that it can provide documents via the World Wide Web. Once the Web server receives a request, it springs into the action. Depending on the type of request, the Web server might look for a Web page, or it might execute a program on the server. Either way, it will always return the results to the Web browser. Even if it cannot process a request, an error page is sent to the browser. Currently the leading Web servers are Apache Web Server and Microsoft Information Server (Netcraft, 2001)

3. Sending a Request to the Web Server

In order for the Web server to respond to and to execute a server program, the Web browser must package up the user data and issue an HTTP request to the Web server. An HTTP request consists of the URL for the page. A request is typically generated by the browser, but we believe that it is still important to understand how a request is constructed and used.

Each request must specify which method the request is to use. The three most important methods are HEAD, GET and, POST (Ayers, 2000)

The HEAD method simply retrieves information about a document and not the document itself. This method is often used to determine if a hypertext link is valid or to determine when a link was last modified.

The GET and POST methods are used to issue requests to execute a Web program. Even though both of their functions seem to be similar, some major differences between them exist.

The GET method is used if we want to retrieve some information from the server. POST is used whenever we want to send some information to change the contents of a file or database. GET requests are appended to the URL of the Web page. So the amount of data that can be sent to the server is limited. But the POST method allows us to send

more data. The GET method is also used as a default method for all Web browsers. Whenever we write an URL for a Web page request, the browser issues the request as a GET request.

4. Executing the Server Program

Whenever a request for a specific program comes to the server, it determines the necessary operating environment's type. Web servers establish this task through mapping by looking at the extension of the requested file (or the directory in which the file is located). At the configuration phase of the Web server, how to handle specific file types is defined. For example, typically anything in the cgi-bin directory will be treated as a Common Gateway Interface (CGI) script, anything with a .jsp extension will be treated as a Java Server Page, or any file with the extension htm will be directly returned to the browser without any runtime environment.

Once the type of the requested file is determined, the server then loads any required runtime environment for the file to be executed. The required runtime environment is dependent on the Web server and the technology. By directing the request to the right place, the Web server fulfills its responsibility.

5. Sending Back the Results to the Browser

The final step in a Web application is to make some kind of response to the operation and return that response to the browser. Mostly, the executed program (or script) specifies the content type and then writes the response to an output stream. When the Web browser receives the response, it will first look at the response header and determine the mime type so that it knows how to deal with the data. The most common type is "text/html", but other types such as XML, unformatted text, GIFs and even stream audio, can also be sent by the Web server.

APPENDIX C. EAMS DATABASE RELATIONSHIP DIAGRAM AND SQL DDL CODE

1. Relationship Diagram for EAMS Database

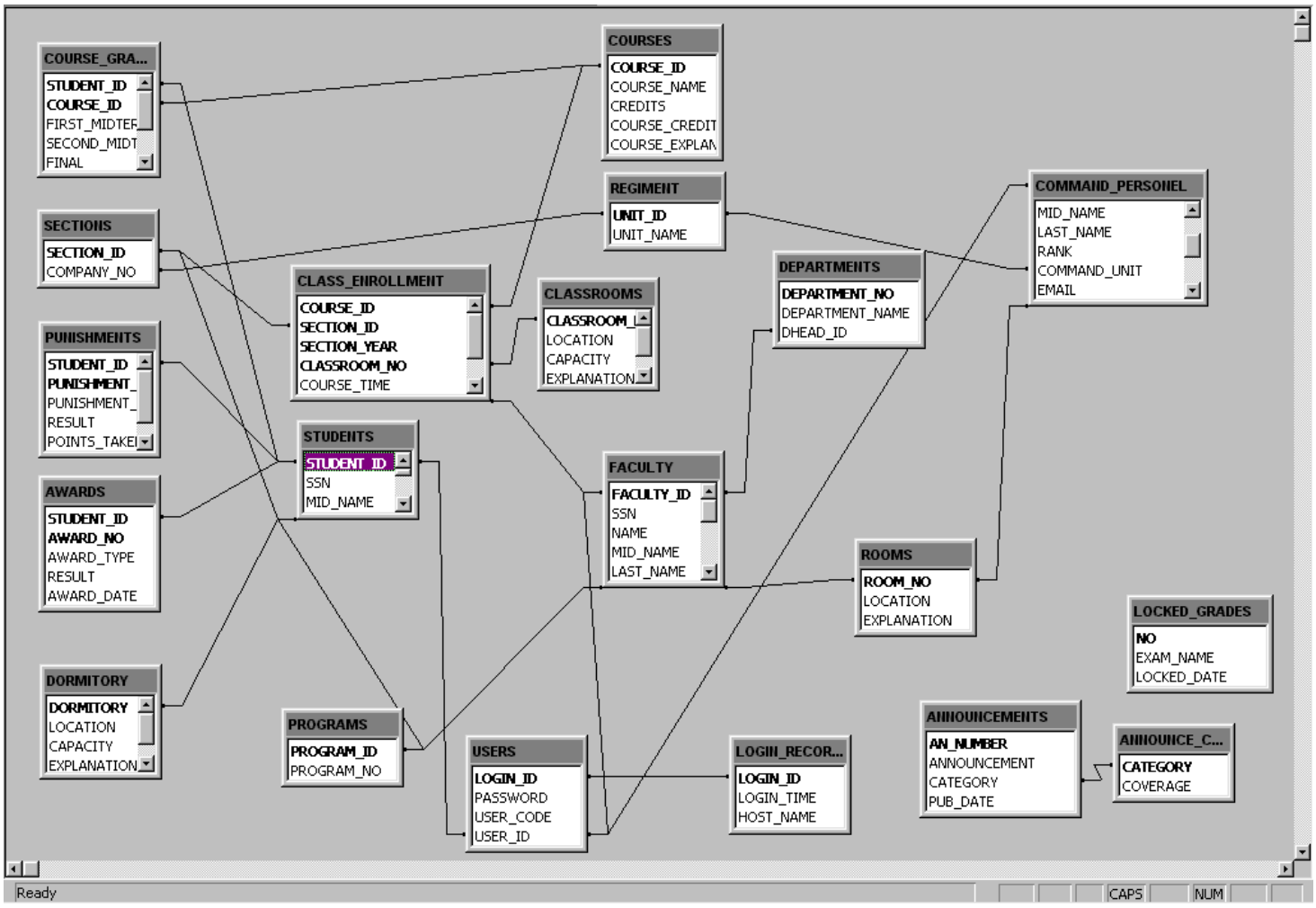

Figure 56. EAMS Database Relationship Diagram

2. SQL DDL Code for EAMS Database

```
CREATE TABLE USERS
(
        LOGIN_ID VARCHAR(10) NOT NULL,
        PASSWORD VARCHAR(10) NOT NULL,
```

```
        USERCODE VARCHAR(10) NOT NULL,
        USER_ID VARCHAR(10) NOT NULL,
        CONSTRAINT PK_USERS PRIMARY KEY (LOGIN_ID)
);

CREATE TABLE STUDENTS
(
STUDENT_ID NUMBER NOT NULL,
   SSN VARCHAR(10) NOT NULL,
   NAME VARCHAR(10) NOT NULL,
   MID_NAME VARCHAR(10) NOT NULL,
   LAST_NAME VARCHAR(10) NOT NULL,
   GENDER CHAR(1) NOT NULL,
   SECTION_ID VARCHAR(10) NOT NULL,
   PROGRAM_ID VARCHAR(10) NOT NULL,
   BIRTHDATE DATE NOT NULL,
   YEAR NUMBER NOT NULL,
   ADDRESS VARCHAR(40) NOT NULL,
   CITY VARCHAR(10) NOT NULL,
   PROVINCE VARCHAR(10) NOT NULL,
   FATHER_NAME VARCHAR(10) NOT NULL,
   REMAINING_POINTS NUMBER(3) NOT NULL,
   EMAIL VARCHAR(30) NOT NULL,
   DORMITORY VARCHAR(10) NOT NULL,
   CONSTRAINT PK_STUDENTS PRIMARY KEY (STUDENT_ID)
   CONSTRAINT FL_STUDENTS_1
      FOREIGN KEY (SECTION_ID)  REFERENCES SECTIONS
   CONSTRAINT FL_STUDENTS_2
      FOREIGN KEY (PROGRAM_ID)  REFERENCES PROGRAMS
   CONSTRAINT FL_STUDENTS_3
      FOREIGN KEY (DORMITORY) REFERENCES DORMITORY

);

CREATE TABLE SECTIONS
(
   SECTION_ID VARCHAR(2) NOT NULL,
   COMPANY_NO NUMBER(4) NOT NULL,
   CONSTRAINT PK_SECTIONS PRIMARY KEY (SECTION_ID)
   CONSTRAINT FL_SECTIONS_1
      FOREIGN KEY (COMPANY_NO)  REFERENCES REGIMENT (UNIT_ID)

);

CREATE TABLE ROOMS
(
```

```
    ROOM_NO VARCHAR(10) NOT NULL,
    LOCATION VARCHAR(40) NOT NULL,
    EXPLANATION VARCHAR(10) NOT NULL,
    CONSTRAINT PK_ROOMS PRIMARY KEY (ROOM_NO)
);

CREATE TABLE REGIMENT
(
    UNIT_ID NUMBER(3) NOT NULL,
    UNIT_NAME VARCHAR(40) NOT NULL,
    CONSTRAINT PK_REGIMENT PRIMARY KEY (UNIT_ID)
);

CREATE TABLE PUNISHMENTS
(
    STUDENT_ID NUMBER(4) NOT NULL,
    PUNISHMENT_NO NUMBER(3) NOT NULL,
    PUNISHMENT_TYPE VARCHAR(100) NOT NULL,
    RESULT VARCHAR(50) NOT NULL,
    POINTS_TAKEN NUMBER(2) NOT NULL,
    PUNISHMENT_DATE DATE NOT NULL
    CONSTRAINT PK_PUNISHMENTS PRIMARY KEY
(STUDENT_ID,PUNISHMENT_NO)
    CONSTRAINT FL_PUNISHMENTS_1
        FOREIGN KEY (STUDENT_ID) REFERENCES STUDENTS

);

CREATE TABLE PROGRAMS
(
    PROGRAM_ID VARCHAR(15) NOT NULL,
    PROGRAM_NO VARCHAR(60) NOT NULL,
    CONSTRAINT PK_PROGRAMS PRIMARY KEY (PROGRAM_ID)
);

CREATE TABLE LOGIN_RECORDS
(
    LOGIN_ID VARCHAR(10) NOT NULL,
    LOGIN_TIME VARCHAR(40) NOT NULL,
    HOST_NAME VARCHAR(90) NOT NULL,
    CONSTRAINT PK_LOGIN_RECORDS PRIMARY KEY (LOGIN_ID,
LOGIN_TIME)
    CONSTRAINT FL_LOGIN_RECORDS_1
        FOREIGN KEY (LOGIN_ID) REFERENCES USERS

);
```

```
CREATE TABLE LOCKED_GRADES
(
   NO NUMBER NOT NULL,
   EXAM_NAME VARCHAR(20) NOT NULL,
   LOCKED_DATE DATE NOT NULL,
   CONSTRAINT PK_LOCKED_GRADES PRIMARY KEY (NO)
);

CREATE TABLE FACULTY
(
   FACULTY_ID VARCHAR(10) NOT NULL,
   SSN VARCHAR(10) NOT NULL,
   NAME VARCHAR(10) NOT NULL,
   MID_NAME VARCHAR(10) NOT NULL,
   LAST_NAME VARCHAR(10) NOT NULL,
   DEGREE VARCHAR(30) NOT NULL,
   PROGRAM_ID VARCHAR(10) NOT NULL,
   RANK VARCHAR(10) NOT NULL,
   EMAIL VARCHAR(30) NOT NULL,
   ROOM_NO VARCHAR(10) NOT NULL,
   TEL-NO VARCHAR(10) NOT NULL,
   ADDRESS VARCHAR(100) NOT NULL,
   CITY VARCHAR(50) NOT NULL,
   PROVINCE VARCHAR(50) NOT NULL,
   CONSTRAINT PK_FACULTY PRIMARY KEY (FACULTY_ID)
   CONSTRAINT FL_FACULTY_1
      FOREIGN KEY (PROGRAM_ID) REFERENCES PROGRAMS
   CONSTRAINT FL_FACULTY_2
      FOREIGN KEY (ROOM_NO) REFERENCES ROOMS
);

CREATE TABLE DORMITORY
(
   DORMITORY_NO VARCHAR(10) NOT NULL,
   LOCATION VARCHAR(40) NOT NULL,
   CAPACITY NUMBER(2) NOT NULL,
   EXPLANATION VARCHAR(90) NOT NULL,
   CONSTRAINT PK_DORMITORY  PRIMARY KEY (DORMITORY_NO)
);

CREATE TABLE DEPARTMENTS
(
   DEPARTMENT_NO NUMBER(2) NOT NULL,
   DEPARTMENT_NAME VARCHAR(40) NOT NULL,
   DHEAD_ID VARCHAR(10) NOT NULL,
```

```
    CONSTRAINT PK_DEPARTMENTS PRIMARY KEY (DEPARTMENT_NO)
    CONSTRAINT FK_DEPARTMENTS_1
       FOREIGN KEY (DHEAD_ID) REFERENCES FACULTY (FACULTY_ID)
);

CREATE TABLE COURSES
(
    COURSE_ID VARCHAR(6) NOT NULL,
    COURSE_NAME VARCHAR(70) NOT NULL,
    CREDITS VARCHAR(7) NOT NULL,
    COURSE_CREDIT NUMBER (2) NOT NULL,
    COURSE_EXPLANATION VARCHAR(4000) NOT NULL,
    CONSTRAINT PK_COURSES PRIMARY KEY (COURSE_ID)
);

CREATE TABLE COURSE_GRADES
(
    STUDENT_ID NUMBER(4) NOT NULL,
    COURSE_ID VARCHAR(10) NOT NULL,
    FIRST_MIDTERM NUMBER(3) NOT NULL,
    SECOND_MIDTERM NUMBER(3) NOT NULL,
    FINAL NUMBER(3) NOT NULL,
    YEAR_TAKEN NUMBER(6) NOT NULL,
    COURSE_GRADE VARCHAR(2) NOT NULL,
    SEMESTRE_TAKEN NUMBER(2) NOT NULL,
    CONSTRAINT PK_COURSE_GRADES PRIMARY KEY (STUDENT_ID,
COURSE_ID, YEAR_TAKEN)
    CONSTRAINT FK_COURSE_GRADES_1
       FOREIGN KEY (STUDENT_ID) REFERENCES STUDENTS
    CONSTRAINT FK_COURSE_GRADES_2
       FOREIGN KEY (COURSE_ID) REFERENCES COURSES
);

CREATE TABLE COMMAND_PERSONEL
(
    PERSON_ID VARCHAR(10) NOT NULL,
    SSN VARCHAR(10) NOT NULL,
    NAME VARCHAR(10) NOT NULL,
    MID_NAME VARCHAR(10) NOT NULL,
    LAST_NAME VARCHAR(10) NOT NULL,
    RANK VARCHAR(10) NOT NULL,
    COMMAND_UNIT NUMBER(4) NOT NULL,
    EMAIL VARCHAR(30) NOT NULL,
    ROOM_NO VARCHAR(10) NOT NULL,
    TEL-NO VARCHAR(10) NOT NULL,
    ADDRESS VARCHAR(100) NOT NULL,
```

```
    CITY VARCHAR(50) NOT NULL,
    PROVINCE VARCHAR(50) NOT NULL,
    CONSTRAINT PK_COMMAND_PERSONEL PRIMARY KEY (PERSON_ID)
    CONSTRAINT FK_COMMAND_PERSONEL_1
        FOREIGN KEY (COMMAND_UNIT) REFERENCES REGIMENT
    CONSTRAINT FK_COMMAND_PERSONEL_2
        FOREIGN KEY (ROOM_NO) REFERENCES ROOMS
);

CREATE TABLE CLASSROOMS
(
    CLASSROOM_NO VARCHAR(10) NOT NULL,
    LOCATION VARCHAR(60) NOT NULL,
    CAPACITY NUMBER (3) NOT NULL,
    EXPLANATION VARCHAR(90) NOT NULL,
    CONSTRAINT PK_CLASSROOMS PRIMARY KEY (CLASSROOM_NO)
);

CREATE TABLE CLASS_ENROLLMENT
(
    COURSE_ID VARCHAR(10) NOT NULL,
    SECTION_ID VARCHAR(10) NOT NULL,
    SECTION_YEAR VARCHAR(10) NOT NULL,
    CLASSROOM_NO VARCHAR(10) NOT NULL,
    COURSE_TIME NUMBER(10)
    COURSE_DATE VARCHAR(10) NOT NULL,
    FACULTY_ID VARCHAR(10) NOT NULL,
    CONSTRAINT PK_CLASS_ENROLLMENT PRIMARY KEY (COURSE_ID,
SECTION_ID, SECTION_YEAR, CLASSROOM_NO)
    CONSTRAINT FK_CLASS_ENROLLMENT_1
        FOREIGN KEY (COURSE_ID) REFERENCES COURSES
    CONSTRAINT FK_CLASS_ENROLLMENT_2
        FOREIGN KEY (SECTION_ID) REFERENCES SECTIONS
    CONSTRAINT FK_CLASS_ENROLLMENT_3
        FOREIGN KEY (CLASSROOM_NO) REFERENCES CLASSROOMS
    CONSTRAINT FK_CLASS_ENROLLMENT_4
        FOREIGN KEY (COURSE_ID) REFERENCES COURSES
    CONSTRAINT FK_CLASSROOMS
        FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY
);

CREATE TABLE AWARDS
(
    STUDENT_ID VARCHAR(10) NOT NULL,
    AWARD_NO NUMBER(2) NOT NULL,
    AWARD_TYPE VARCHAR(60) NOT NULL,
```

```
    RESULT VARCHAR(60) NOT NULL,
    AWARD_DATE DATE NOT NULL,
    CONSTRAINT PK_AWARDS PRIMARY KEY (STUDENT_ID, AWARD_NO)
    CONSTRAINT FK_AWARDS_1
        FOREIGN KEY (STUDENT_ID) REFERENCES STUDENTS
);

CREATE TABLE ANNOUNCEMENTS
(
    AN_NUMBER NUMBER(10) NOT NULL,
    ANNOUNCEMENT VARCHAR(4000) NOT NULL,
    CATEGORY NUMBER(10) NOT NULL,
    PUB_DATE VARCHAR(20) NOT NULL,
    CONSTRAINT PK_ANNOUNCEMENTS PRIMARY KEY (AN_NUMBER)
);

CREATE TABLE ANNOUNCE_CAT
(
    CATEGORY NUMBER(10) NOT NULL,
    COVERAGE VARCHAR(10) NOT NULL,
    CONSTRAINT PK_ANNOUNCE_CAT PRIMARY KEY (CATEGORY_NUMBER)
);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. SOURCE CODE OF THE IMPLEMENTATION

1. Source Code of the Academy Servlet

```
/**
 * Title: Academy.java
 * Description: The main servlet that has all the application logic for the system
 * @author : Rasim Topuz, Ltjg, Turkish Navy
 * @version 1.0, February 2002
 */

package webadmin;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;

/*
Class Decleration for Academy
*/

public class Academy extends HttpServlet
{

  private PoolManager poolMgr;
  private LogWriter logWriter;


/*
Class initilization for Academy
*/
  public void init() throws ServletException
  {
    //this is the connection pool manager that will be
    //used for the whole application
    poolMgr = PoolManager.getInstance();

    //this is the log file to record all the activities done by servlet
    PrintWriter pw = new PrintWriter(System.err,true);//
    logWriter = new LogWriter("AcademyServlet",2,pw);
    try{
    pw = new PrintWriter(new FileWriter("ServletActivityLogFile", true), true);
    }
    catch(IOException e){
   logWriter.log(e,"servlet initilized",1);

    }
    logWriter.setPrintWriter(pw);
    logWriter.log("servlet initilized",2);//2 means info for the log
  }

/*
```

```
Method doPost: Used for handling user requests
*/

  public void doPost(HttpServletRequest req, HttpServletResponse res)
              throws IOException, ServletException
  {
    getTheParameters(req,res);
  }

/*
Class destroy method for Academy
*/
  public void destroy()
  {
    poolMgr.release();
    super.destroy();
  }

/*
Method getTheParameters: handles the request.
*/
  public void getTheParameters(HttpServletRequest request,HttpServletResponse respond){
    int count = 0;
    Hashtable myParameterValues = new Hashtable();
    Enumeration paramNames = request.getParameterNames();
    String value = "";
    //load all parameters into a hashtable.
    while(paramNames.hasMoreElements()) {
      String paramName = (String)paramNames.nextElement();
      count++;
      logWriter.log("post request & parameters received from " + request.getRemoteAddr(),2);
      String[] paramValues =
      request.getParameterValues(paramName);
      if (paramValues.length == 1) {
        String paramValue = paramValues[0];
        if (paramValue.length() == 0){
          //no value
        } else{
          value = paramValue;
        }
      } else {
        for(int i=0; i<paramValues.length; i++) {

          value = value + paramValues[i].toString();
        }//end of for

      }//end of if
    Object val = myParameterValues.put(paramName,value);
    }//end of while

    //this part handles the user request according to the action parameter
    Object val = myParameterValues.get("action");
    if (val != null) {
      if (val.toString().equals("LOGIN")){
          executeLogin(myParameterValues,request,respond);
      } else if (val.toString().equals("courses")){
```

```
            executeCourses(myParameterValues, request, respond);
        } else if (val.toString().equals("home_page")){
            executeHomePage(myParameterValues, request, respond);
        } else if (val.toString().equals("schedule")){
            executeSchedule(myParameterValues, request, respond);
        } else if (val.toString().equals("personal_update")){
            executeUpdate(myParameterValues, request, respond);
        } else if (val.toString().equals("studentsForProf")){
            executeStudentsForProf(myParameterValues, request, respond);
        } else if (val.toString().equals("gradesForStudent")){
            executeGradesForStudent(myParameterValues, request, respond);
        } else if (val.toString().equals("gradeUpdate")){
            executeGradeUpdate(myParameterValues, request, respond);
        } else if (val.toString().equals("change_Password")){
            executeChangePassword(myParameterValues, request, respond);
        } else if (val.toString().equals("companyInfoForStudent")){
            executeCompanyInfoForStudent(myParameterValues, request, respond);
        } else if (val.toString().equals("department")){
            executeDepartment(myParameterValues, request, respond);
        } else if (val.toString().equals("studentsForDepHead")){
            executeStudentsForDepHead(myParameterValues, request, respond);
        } else if (val.toString().equals("unitsForRegiment")){
            executeUnitsForRegiment(myParameterValues, request, respond);
        } else if (val.toString().equals("showCommanderInfo")){
            executeShowCommanderInfo(myParameterValues, request, respond);
        } else if (val.toString().equals("showCompanyInfo")){
            executeShowCompanyInfo(myParameterValues, request, respond);
        } else if (val.toString().equals("showSectionInfo")){
            executeSectionInfoForRegiment(myParameterValues, request, respond);
        } else if (val.toString().equals("showStudentInfoForRegiment")){
            executeStudentInfoForRegiment(myParameterValues, request, respond);
        } else if (val.toString().equals("personal_information")){ //personal information
            executePersonal(myParameterValues, request, respond);
        } else if (val.toString().equals("log_off")){ //personal information
            executeLogOff(myParameterValues, request, respond);
        } else {
        logWriter.log("Unknown action requested from " + request.getRemoteAddr(),1);
        }  //end of if
    }//end of if
  }//end of getParameters()

/*
  Method executeLogOff:executes the log Off sequence for theuser, and sends a
  log off jsp file
  */
   public void executeLogOff(Hashtable myParameterValues,
                        HttpServletRequest request,
                        HttpServletResponse respond){

     UserType myUser = new UserType();
     myUser = checkCookie(request,respond);
     logWriter.log("log off sequence requested by " + myUser.myLoginId +
                 " from " + request.getRemoteAddr(),2);
     if (myUser.state) {//the result of the cookiecheck must be affirmative.
         Communicator myLogOff;
          myLogOff = new Communicator();
```

```
        doLogOff(request, respond);

          java.util.Date myDate = new java.util.Date();
           myLogOff.setTime(myDate.toString());
           myLogOff.setName(myUser.myName + " " +myUser.myLastName);

           HttpSession session = request.getSession(true);
           session.putValue("myLogOff",myLogOff);
           gotoPage("/src/webadmin/LogOff.jsp",request,respond);
    }else {
      logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
            " at " + request.getRemoteAddr(),1);
    }//end of if
  }//end of execute log off.



  /*
  Method checkCookie:checks if the current request and the cookies are the same.
  */

  private void doLogOff(HttpServletRequest request,
                  HttpServletResponse response) {
    Cookie[] myCookie = request.getCookies();
    UserType myUser = new UserType();
    //String user = "";
    String userTime = "";
    for(int i=0; i<myCookie.length; i++) {
       Cookie cookie = myCookie[i];
       if (cookie.getName().equals("loginId")){
         myUser.myLoginId = cookie.getValue();
       }

     }//end of for



//delete the login information from the database
       Connection conn = poolMgr.getConnection("myThesis");
       String user = myUser.myLoginId;
       boolean result = false;
      if (conn == null)
      {
          logWriter.log("database connection error",1);
      }

      ResultSetMetaData md = null;
      Statement stmt = null;
      try
      {
        stmt = conn.createStatement();
        StringBuffer myQuery = new StringBuffer();
        myQuery.append("delete from login_records where login_id='");
        myQuery.append(user + "'");
        stmt.executeUpdate(myQuery.toString());
```

```
        stmt.close();
        poolMgr.freeConnection("myThesis", conn);
      }//end of try
      catch (SQLException e)
      {
        logWriter.log(e," Exception error",1);
      }

    }//end of do log off

/*
Method executeStudentInfo:executes StudentInfo request for regiment personnel
*/

  public void executeStudentInfoForRegiment(Hashtable myParameterValues,
                        HttpServletRequest request,
                        HttpServletResponse respond){
      UserType myUser = new UserType();
      myUser = checkCookie(request,respond);


      logWriter.log("Student info for regiment requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);

      if (myUser.state) {//the result of the cookiecheck must be affirmative.
         Communicator myFaculty;
         myFaculty = new Communicator();
         //--connect to the db and get+arrange the courses for the user.
         java.util.Date myDate = new java.util.Date();
         myFaculty = getStudentInfoForRegiment(myParameterValues,myUser);
         myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
         myFaculty.setLastName(myUser.myLastName);
         myFaculty.setOperation("showStudentInformation");
         myFaculty.setTime(myDate.toString());
         HttpSession session = request.getSession(true);
         session.putValue("myRegiment",myFaculty);
         logWriter.log("Student info for regiment sent to " + myUser.myLoginId +
             " at " + request.getRemoteAddr(),2);
         gotoPage("/src/webadmin/Regiment.jsp",request,respond);
      } else {
       logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
        }//end of if
    }//end of execute student info for regiment.

/*
Method getStudentInfoForRegiment: gets the data from the database and sends
the data to the proper jsp file.
*/

    public Communicator getStudentInfoForRegiment(Hashtable myParameterValues,
                                    UserType myUser){
      Connection conn = poolMgr.getConnection("myThesis");
      String announcements = new String();
      Communicator myRegiment = new Communicator();
```

```
int commandUnit = 0;
boolean result = false;
  if (conn == null)
  {
  logWriter.log("database connection error",1);
    //return false;
  }
  ResultSet rs = null;
  ResultSetMetaData md = null;
  Statement stmt = null;

  Object val = myParameterValues.get("student_id");
  String studentId = val.toString().trim();

  try
  {
    stmt = conn.createStatement();

  StringBuffer myQuery = new StringBuffer();
  myQuery.append(" select u.login_id ");
  myQuery.append(" from users u, students s ");
  myQuery.append(" where u.user_id = s.student_id and ");
  myQuery.append(" s.student_id='" + studentId + "'");
  rs = stmt.executeQuery(myQuery.toString());
  rs.next();
  String studentLogin = rs.getString(1);

  myRegiment = getStudentPersonal(studentLogin);
  myRegiment.setName1(myRegiment.getName());
  myRegiment.setLastName1(myRegiment.getLastName());
  myRegiment.setLogin1(myRegiment.getLogin());

  StringBuffer awardInfo = new StringBuffer();
  awardInfo.append(" select distinct s.student_id, ");
  awardInfo.append(" a.award_no,a.award_type,a.result,a.award_date");
  awardInfo.append(" from students s, awards a, users u ");
  awardInfo.append(" where s.student_id = u.user_id and ");
  awardInfo.append(" s.student_id = a.student_id and ");
  awardInfo.append(" u.login_id = ");

  StringBuffer punishmentInfo = new StringBuffer();
  punishmentInfo.append(" select distinct s.student_id, ");
  punishmentInfo.append(" p.punishment_no,p.punishment_type, ");
  punishmentInfo.append(" p.result,p.points_taken,p.punishment_date ");
  punishmentInfo.append(" from students s, punishments p, users u ");
  punishmentInfo.append(" where s.student_id = u.user_id and ");
  punishmentInfo.append(" s.student_id = p.student_id and ");
  punishmentInfo.append(" u.login_id = ");

   awardInfo.append("'" + studentLogin + "'");

   rs = stmt.executeQuery(awardInfo.toString());
   String award = "";
   while(rs.next()){
    rs.getString(1);
   award += " <tr> "+
```

```
        "<td width=\"84\" height=\"19\">"+
        rs.getString(2) + "</td>"+
        "<td width=\"98\" height=\"19\">"+
        rs.getString(3) + "</td>"+
        "<td width=\"248\" height=\"19\">"+
        rs.getString(4) + "</td>"+
        "<td width=\"231\" height=\"19\" colspan=\"2\">"+
        rs.getString(5).substring(0,11) + "</td>"+
        " </tr>";
}
myRegiment.setAwards(award);
punishmentInfo.append("'" + studentLogin + "'");

rs = stmt.executeQuery(punishmentInfo.toString());
String punishments = "";
while(rs.next()){
 rs.getString(1);
punishments += " <tr> "+
        "<td width=\"84\" height=\"19\">"+
        rs.getString(2) + "</td>"+
        "<td width=\"98\" height=\"19\">"+
        rs.getString(3) + "</td>"+
        "<td width=\"248\" height=\"19\">"+
        rs.getString(4) + "</td>"+
        "<td width=\"126\" height=\"19\"> "+
        rs.getString(5) + "</td>"+
        "<td width=\"99\" height=\"19\" >"+
        rs.getString(6).substring(0,11) + "</td>"+
        " </tr>";

}
myQuery = new StringBuffer();
myQuery.append("select remaining_points ");
myQuery.append("from students ");
myQuery.append("where student_id = '" + studentId + "'");
rs= stmt.executeQuery(myQuery.toString());
rs.next();
punishments += " <tr> "+
        "<td width='400' colspan='3' height='19'>"+
         "</td>"+
        "<td width=\"126\" height=\"19\">"+
        "Current Remaining Points" + "</td>"+
        "<td width=\"99\" height=\"19\">"+
        rs.getString(1) + "</td>"+
        " </tr>";

myRegiment.setPunishments(punishments);
String year = "2002";//current year and semester was assumed here
String semesterValue = "2";

 myQuery = new StringBuffer();
//-------------
myQuery.append(" select distinct f.name,f.last_name, c.course_id, ");
myQuery.append(" co.course_name,g.first_midterm, ");
myQuery.append(" g.second_midterm,g.final,g.course_grade, ");
myQuery.append(" f.email,co.credits,co.course_explanation ");
```

```
    myQuery.append(" from class_enrollment c, faculty f, ");
    myQuery.append(" students s, course_grades g,users u, ");
    myQuery.append(" courses co ");
    myQuery.append(" where u.user_id = s.student_id and ");
    myQuery.append(" s.student_id = g.student_id and ");
    myQuery.append(" s.section_id = c.section_id and ");
    myQuery.append(" c.course_id = g.course_id and ");
    myQuery.append(" c.faculty_id = f.faculty_id and ");
    myQuery.append(" u.login_id = '" + studentLogin + "' and ");
    myQuery.append(" c.course_id = co.course_id and ");
    myQuery.append(" g.year_taken = '" + year + "' and ");
    myQuery.append(" g.semestre_taken = '" + semesterValue + "'");


    rs = stmt.executeQuery(myQuery.toString());
    String fname = "";
    String fLname = "";
    String cId = "";
    String cName = "";
    String firstExam = "";
    String secondExam = "";
    String finalExam = "";
    String courseGrade = "";
    String fmail = "";
    String courseExp = "";
    String courseCredit = "";
    String fId = "";

    String grades = "";

     while (rs.next())
     {
    fname = rs.getString(1);
    fLname =rs.getString(2);
    cId = rs.getString(3);
    cName = rs.getString(4);
    firstExam = rs.getString(5);
    secondExam = rs.getString(6);
    finalExam = rs.getString(7);
    courseGrade = rs.getString(8);
    fmail = rs.getString(9);
    fId =  fname.substring(0,1).concat(fLname);
    courseExp = rs.getString(10);
    courseCredit = rs.getString(11);
    String profName = fname + " " + fLname;
    for(int i=(11-profName.length());i>=0;i--){
       profName += " ";
    }

    grades += "<tr>"+
"<td width=\"86\" height=\"22\" bgcolor=\"#C0C0C0\" valign=\"middle\">" +
"<p align=\"center\">"+
"<INPUT onclick=\"javascript:facultyview('" + fname +
"','" + fLname + "','" + fId + "','" + fmail + "',200,200)\" type=button value=\""+profName + "\">"+
"</p>"+
"</td>" +
```

```
  "<td width=\"108\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<p align=\"center\">"+
  "<INPUT onclick=\"javascript:courseview('" + cId +
  "','" + cName + "','" + courseExp + "','" + courseCredit + "',400,300)\" type=button value=\""+cId +
"\">"+
  "</p>"+
  "</td>" +
  "<td width=\"94\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">"+cName +"</font></td>"+
  "<td width=\"106\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">" + firstExam + "</font></td>"+
  "<td width=\"125\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">" + secondExam + "</font></td>"+
  "<td width=\"63\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">" + finalExam + "</font></td>"+
  "<td width=\"94\" height=\"22\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">" + courseGrade + "</font></td>"+
  "  </tr>";
       }//end of while

     myRegiment.setCourses(grades);
     myUser = new UserType();
     myUser.myLoginId = studentLogin;
     myRegiment.setSchedule(getSchedule(myUser));
     rs.close();
     stmt.close();
    }//end of try
    catch (Exception e)
    {
     logWriter.log(e,"database connection error",1);
    }
    finally
    {
     poolMgr.freeConnection("myThesis", conn);
     logWriter.log("student info retrieved",2);
     return myRegiment;
    }

 }//end of getsectionInfo

/*
Method executeSectionInforForRegiment: executes section info for regiment.
*/

public void executeSectionInfoForRegiment(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);

   logWriter.log("section info for regiment requested by " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.

      Communicator myFaculty;
```

```
        myFaculty = new Communicator();
       //--connect to the db and get+arrange the courses for the user.
          java.util.Date myDate = new java.util.Date();
          myFaculty = getSectionInfoForRegiment(myParameterValues,myUser);
          myFaculty.setLogin(myUser.myLoginId);
          myFaculty.setName(myUser.myName);
          myFaculty.setLastName(myUser.myLastName);
          myFaculty.setOperation("sectionInfoForRegiment");
          myFaculty.setTime(myDate.toString());
          HttpSession session = request.getSession(true);
          session.putValue("myRegiment",myFaculty);
          gotoPage("/src/webadmin/Regiment.jsp",request,respond);
    }else {
     logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
      }//end of if
  }//end of execute units for regiment.

/*
Method getSectionInfoForRegiment: gets the data from the database and sends
the data to the proper jsp file.
*/

  public Communicator getSectionInfoForRegiment(Hashtable myParameterValues,UserType myUser){
  Connection conn = poolMgr.getConnection("myThesis");
  String announcements = new String();
  Communicator myRegiment = new Communicator();
  int commandUnit = 0;
  boolean result = false;
    if (conn == null)
    {
     logWriter.log("database connection error",1);

    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;



    Object val = myParameterValues.get("section");
    String section = val.toString().trim();
    myRegiment.setSection(section);


    try
    {
      stmt = conn.createStatement();
    //first get the necessary info about the commander
          StringBuffer myQuery = new StringBuffer();
      myQuery.append(" select r.rank,r.name,r.last_name, ");
      myQuery.append(" r.command_unit,c.unit_name ");
      myQuery.append(" from command_personel r, users u, regiment c ");
      myQuery.append(" where u.login_id = '" + myUser.myLoginId + "' and ");
      myQuery.append(" u.user_id = r.person_id and ");
      myQuery.append(" r.command_unit = c.unit_id ");
```

```
   rs= stmt.executeQuery(myQuery.toString());
   rs.next();
   myRegiment.setRegComName(rs.getString(1) + " " +
                        rs.getString(2) + " " +
                        rs.getString(3));

   myQuery = new StringBuffer();
   myQuery.append(" select r.unit_name ");
   myQuery.append(" from sections s, regiment r ");
   myQuery.append(" where s.section_id = '" + section + "' and ");
   myQuery.append(" s.company_no = r.unit_id ");
   rs = stmt.executeQuery(myQuery.toString());
   rs.next();
   myRegiment.setCompanyName(rs.getString(1));

   myQuery = new StringBuffer();
   myQuery.append(" select student_id,name,last_name,remaining_points ");
   myQuery.append(" from students ");
   myQuery.append(" where section_id = '" + section + "' ");
   myQuery.append(" order by student_id ");

   rs= stmt.executeQuery(myQuery.toString());
  String regiment = "";
  String sId = "";
  int count = 0;
   while(rs.next()){
    count++;
    sId= rs.getString(1);
   regiment +=  "<tr>"+"<td width='20%' height='1'>"+
   "<form method='POST' style='word-spacing: 0; margin: 0' "+
   " action='/servlet/webadmin.Academy' onSubmit=">"+
   " <p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='"+sId+"'name='student_id'></p>"+
   "<input type='hidden' name='action' value='showStudentInfoForRegiment'>"+
   "</form></td>"+
   "<td width='20%' height='1'>" + rs.getString(2) + "</td>"+
   "<td width='20%' height='1' colspan='2'>"+ rs.getString(3) +"</td>"+
   "<td width='20%' height='1'>"+ rs.getString(4) +"</td>  </tr>";
    }//end of while
    myRegiment.setAwards(String.valueOf(count));
    myRegiment.setRegiment(regiment);

   rs.close();
   stmt.close();
}//end of try
catch (Exception e)
{
    logWriter.log(e," Exception error",1);
}
finally
{
   poolMgr.freeConnection("myThesis", conn);
   return myRegiment;
}
```

```
}//end of getsectionInfo


/*
Method executeShowCompanyInfo:executes CompnayInfo request for the regiment personnel
*/

 public void executeShowCompanyInfo(Hashtable myParameterValues,
                     HttpServletRequest request,
                     HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);
    logWriter.log("company info for regiment requested by " + myUser.myLoginId +
              " from " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.

      Communicator myFaculty;
      myFaculty = new Communicator();
     //--connect to the db and get+arrange the courses for the user.
        java.util.Date myDate = new java.util.Date();
        myFaculty = getCompanyInfo(myParameterValues,myUser);
        myFaculty.setLogin(myUser.myLoginId);
        myFaculty.setName(myUser.myName);
        myFaculty.setLastName(myUser.myLastName);
        myFaculty.setOperation("showCompanyInfo");
        myFaculty.setTime(myDate.toString());
        HttpSession session = request.getSession(true);
        session.putValue("myRegiment",myFaculty);
        gotoPage("/src/webadmin/Regiment.jsp",request,respond);

   }else {
    logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),1);
     }//end of if
}//end of execute company info for regiment.

 /*
Method getCompanyInfo: gets the data from the database and sends
the data to the proper jsp file.
*/

 public Communicator getCompanyInfo(Hashtable myParameterValues,UserType myUser){
   Connection conn = poolMgr.getConnection("myThesis");
   String announcements = new String();
   Communicator myRegiment = new Communicator();
   int commandUnit = 0;
   boolean result = false;
    if (conn == null)
    {
       logWriter.log("database connection error",1);
      //return false;
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
```

```
Object val = myParameterValues.get("battalion_no");
Object val2 = new Object();
String companyNo = "";
String temp = "";
if (myParameterValues.containsKey("1")){
  temp = "1";
}else if (myParameterValues.containsKey("2")){
  temp = "2";
}else if (myParameterValues.containsKey("3")){
  temp = "3";
}else if (myParameterValues.containsKey("4")){
  temp = "4";
}else{
  logWriter.log(" Parameter error",1);
}
  companyNo = val.toString() + temp;

try
{
   stmt = conn.createStatement();
//first get the necessary info about the commander
   StringBuffer myQuery = new StringBuffer();
   myQuery.append(" select r.rank,r.name,r.last_name, ");
   myQuery.append(" r.command_unit,c.unit_name ");
   myQuery.append(" from command_personel r, users u, regiment c ");
   myQuery.append(" where u.login_id = '" + myUser.myLoginId + "' and ");
   myQuery.append(" u.user_id = r.person_id and ");
   myQuery.append(" r.command_unit = c.unit_id ");

   rs= stmt.executeQuery(myQuery.toString());
   rs.next();
   myRegiment.setRegComName(rs.getString(1) + " " +
                        rs.getString(2) + " " +
                        rs.getString(3));
   myQuery = new StringBuffer();
   myQuery.append(" select r.person_id,r.rank,r.name,r.last_name, ");
   myQuery.append(" c.unit_name,u.login_id ");
   myQuery.append(" from command_personel r,regiment c,users u ");
   myQuery.append(" where r.command_unit = '" + companyNo + "' and ");
   myQuery.append(" r.command_unit = c.unit_id  and ");
   myQuery.append(" u.user_id = r.person_id ");

   rs= stmt.executeQuery(myQuery.toString());
   rs.next();
   myRegiment.setUserId(rs.getString(1));
   myRegiment.setCompComName(rs.getString(2) + " " +
                        rs.getString(3) + " " +
                        rs.getString(4));

    myRegiment.setCompanyName(rs.getString(5));
    myRegiment.setLogin1(rs.getString(6));

   myQuery = new StringBuffer();
   myQuery.append(" select section_id  ");
   myQuery.append(" from sections ");
```

```
    myQuery.append(" where company_no = '" + companyNo +"'");
    rs= stmt.executeQuery(myQuery.toString());
    rs.next();
    myRegiment.setSection(rs.getString(1));
    rs.next();
    myRegiment.setCourse(rs.getString(1));
    rs.close();
    stmt.close();
   }//end of try
   catch (Exception e)
   {
            logWriter.log(e," Exception error",1);
   }
   finally
   {
    poolMgr.freeConnection("myThesis", conn);
    return myRegiment;
   }

 }//end of getCompanyInfo


/*
Method executeShowCommanderInfo:executes CommanderInfo request for
regiment personnel
*/
public void executeShowCommanderInfo(Hashtable myParameterValues,
                    HttpServletRequest request,
                    HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);
   logWriter.log("commander info for regiment requested by " + myUser.myLoginId +
            " from " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.
      Communicator myFaculty;

      Object val = myParameterValues.get("person_loginid");

      myFaculty = new Communicator();
     //--connect to the db and get+arrange the courses for the user.
        java.util.Date myDate = new java.util.Date();
        myFaculty = getRegimentPersonal(val.toString());
        myFaculty.setLogin(myUser.myLoginId);
        myFaculty.setName(myUser.myName);
        myFaculty.setLastName(myUser.myLastName);
        myFaculty.setOperation("commanderInfoForRegiment");
        myFaculty.setTime(myDate.toString());
        HttpSession session = request.getSession(true);
        session.putValue("myRegiment",myFaculty);
        gotoPage("/src/webadmin/Regiment.jsp",request,respond);

   }else {
    logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
            " at " + request.getRemoteAddr(),1);
    }//end of if
```

```
 }//end of execute shoe commander info.


/*
Method executeUnitsForRegiment:executes units Info request for
regiment personnel
*/

public void executeUnitsForRegiment(Hashtable myParameterValues,
                    HttpServletRequest request,
                    HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);
    logWriter.log("Units info for regiment requested by " + myUser.myLoginId +
              " from " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.
      Communicator myFaculty;
      myFaculty = new Communicator();
     //--connect to the db and get+arrange the courses for the user.
        java.util.Date myDate = new java.util.Date();
        myFaculty = getUnitsForRegiment(myUser);
        myFaculty.setLogin(myUser.myLoginId);
        myFaculty.setName(myUser.myName);
        myFaculty.setLastName(myUser.myLastName);
        myFaculty.setOperation("unitsForRegiment");
        myFaculty.setTime(myDate.toString());
        HttpSession session = request.getSession(true);
        session.putValue("myRegiment",myFaculty);
        gotoPage("/src/webadmin/Regiment.jsp",request,respond);

   }else {
    logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),1);
     }//end of if
 }//end of execute units for regiment.

/*
Method getUnitsInfoForRegiment: gets the data from the database and sends
the data to the proper jsp file.
*/

 public Communicator getUnitsForRegiment(UserType myUser){
   Connection conn = poolMgr.getConnection("myThesis");
   String announcements = new String();
   Communicator myRegiment = new Communicator();
   int commandUnit = 0;
   boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
      //return false;
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
```

```
    try
    {
      stmt = conn.createStatement();
    //first get the necessary info about the commander
      StringBuffer myQuery = new StringBuffer();
      myQuery.append(" select r.person_id,r.rank,r.name,r.last_name, ");
      myQuery.append(" r.command_unit,c.unit_name ");
      myQuery.append(" from command_personel r, users u, regiment c ");
      myQuery.append(" where u.login_id = '" + myUser.myLoginId + "' and ");
      myQuery.append(" u.user_id = r.person_id and ");
      myQuery.append(" r.command_unit = c.unit_id ");
      rs= stmt.executeQuery(myQuery.toString());
      rs.next();
      myRegiment.setUserId(rs.getString(1));

      myRegiment.setRegComName(rs.getString(2) + " " +
                          rs.getString(3) + " " +
                          rs.getString(4));
      commandUnit = rs.getInt(5);
      myRegiment.setCompanyName(rs.getString(6));

    //check what is the command type.
    if (commandUnit == 2){//he is the regiment commander

      myQuery = new StringBuffer();
      myQuery.append(" select c.unit_name,r.rank,r.name,r.last_name, ");
      myQuery.append(" r.person_id, u.login_id , r.command_unit ");
      myQuery.append(" from command_personel r,regiment c,users u ");
      myQuery.append(" where r.command_unit < 15 and ");
      myQuery.append(" r.command_unit > 10 and  ");
      myQuery.append(" r.command_unit = c.unit_id and ");
      myQuery.append(" u.user_id= r.person_id ");
      myQuery.append(" order by r.command_unit ");

      rs = stmt.executeQuery(myQuery.toString());
      int number = 0;
      String regiment = "";
      while (rs.next()){
        number = 0;
        regiment += " <tr> "+
  " <td width='39%' height='19' bgcolor='#0000FF'><b><font color='#FFFFFF'>Battalion name:</font>
</b></td>"+
  " <td width='61%' height='19' bgcolor='#0000FF'><font color='#FFFFFF'>"+
  rs.getString(1) + "</font></td> </tr>  <tr>"+
  " <td width='39%' height='64'><b>Battalion Commander: </b></td>"+
  " <td width='61%' height='64'>"+
  "<form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=">"+
  "     <p style='line-height: 100%; margin: 0'><input type='submit' value='"+
  rs.getString(2) + " " + rs.getString(3) + " " + rs.getString(4) +
  "' name='commander'></p>"+
  "  <input type='hidden' name='person_id' value='"+ rs.getString(5) + "'>"+
  "  <input type='hidden' name='person_loginid' value='"+ rs.getString(6) + "'>"+
  "  <input type='hidden' name='action' value='showCommanderInfo'>"+
```

```
  "   </form>   </td>  </tr>  <tr>"+
  " <td width='39%' height='1'><b>Companies in the battalion:</b></td>"+
  " <td width='61%' height='1'>"+
   "<form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=">"+
   "    <p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='"+ ++number +". Company' name='" + number + "'>"+
   "<input type='submit' value='"+ ++number +". Company' name='" + number + "'>"+
   "</p>       <p style='line-height: 100%; margin: 0'> </p>"+
   "<p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='"+ ++number +". Company' name='" + number + "'>"+
   "<input type='submit' value='"+ ++number +". Company' name='" + number + "'>"+
   "</p>"+
    " <input type='hidden' name='battalion_no' value='" + rs.getString(7) +"'>"+
    " <input type='hidden' name='action' value='showCompanyInfo'>"+
    " </form>     </td>  </tr>";
     }//end of while

     myRegiment.setRegiment(regiment);

     } else if ((commandUnit > 10) && (commandUnit < 15)){//battalion commmander

        String regiment =" <tr> "+
   "<td width='39%' height='19' bgcolor='#0000FF'><b><font color='#FFFFFF'>Battalion name:</font>
</b></td>"+
   "<td width='61%' height='19' bgcolor='#0000FF'><font color='#FFFFFF'>"+
   myRegiment.getCompanyName() + "</font></td>  </tr>  <tr>"+
   "<td width='39%' height='1'><b>Companies in the battalion:</b></td>"+
   " <td width='61%' height='1'>"+
   "<form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=">"+
   "    <p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='1.Company' name='" + 1 + "'>"+
   "<input type='submit' value='2.Company' name='" + 2 + "'>"+
      "</p>       <p style='line-height: 100%; margin: 0'> </p>"+
      "<p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='3.Company' name='" + 3 + "'>"+
   "<input type='submit' value='4.Company' name='" + 4 + "'>"+
      "</p>"+
    " <input type='hidden' name='battalion_no' value='" + commandUnit +"'>"+
    " <input type='hidden' name='action' value='showCompanyInfo'>"+
    " </form>     </td>  </tr>";
     myRegiment.setRegiment(regiment);

     }else if (commandUnit > 110){//Company commmander
        myQuery = new StringBuffer();
        myQuery.append(" select section_id  ");
        myQuery.append(" from sections ");
        myQuery.append(" where company_no = '" + commandUnit +"'");
        rs= stmt.executeQuery(myQuery.toString());
        rs.next();
        String s1 = rs.getString(1);
        rs.next();
        String s2 = rs.getString(1);
        String regiment ="";
        regiment +=" <tr> "+
```

```
   "<td width='39%' height='19' bgcolor='#0000FF'><b><font color='#FFFFFF'>Company name:</font>
</b></td>"+
   "<td width='61%' height='19' bgcolor='#0000FF'><font color='#FFFFFF'>"+
   myRegiment.getCompanyName() + "</font></td>  </tr>  <tr>"+
   "<td width='39%' height='1'><b>Sections in the company:</b></td>"+
   " <td width='61%' height='1'>"+
   "<form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=">"+
   "    <p style='line-height: 100%; margin: 0'>"+
   "<input type='submit' value='            " + s1 + "         ' name='section'><p>"+
   "<input type='submit' value='            " + s2 + "         ' name='section'>"+
    " <input type='hidden' name='action' value='showSectionInfo'>"+
    " </form>     </td>  </tr>";

     myRegiment.setRegiment(regiment);
     }
     rs.close();
     stmt.close();
     }//end of try
     catch (Exception e)
     {
       logWriter.log(e," Exception error",1);
     }
     finally
     {
       poolMgr.freeConnection("myThesis", conn);
       return myRegiment;
     }
  }//end of getUnitsForRegiment


  /*
  Method executeDepartment:executes department info request for dep.head
  */
  public void executeDepartment(Hashtable myParameterValues,
                        HttpServletRequest request,
                        HttpServletResponse respond){

     UserType myUser = new UserType();
     myUser = checkCookie(request,respond);
      logWriter.log("Department info for faculty requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);
     if (myUser.state) {//the result of the cookiecheck must be affirmative.

         Communicator myFaculty;
         myFaculty = new Communicator();
        //--connect to the db and get+arrange the courses for the user.
           java.util.Date myDate = new java.util.Date();
           myFaculty = getDepartmentInfo(myUser);
           myFaculty.setLogin(myUser.myLoginId);
           myFaculty.setName(myUser.myName);
           myFaculty.setLastName(myUser.myLastName);
           myFaculty.setOperation("department");
           myFaculty.setTime(myDate.toString());
           HttpSession session = request.getSession(true);
           session.putValue("myDepHead",myFaculty);
```

```
            gotoPage("/src/webadmin/DepartmentHead.jsp",request,respond);

        }else {
         logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                  " at " + request.getRemoteAddr(),1);
         }//end of if
     }//end of execute courses.

    /*
    Method getDepartmentInfo: gets the data from the database and sends
    the data to the proper jsp file.
    */


     public Communicator getDepartmentInfo(UserType myUser){
       Connection conn = poolMgr.getConnection("myThesis");
       String announcements = new String();
       Communicator myFaculty = new Communicator();
       boolean result = false;
        if (conn == null)
        {
            logWriter.log("database connection error",1);
          //return false;
        }
        ResultSet rs = null;
        ResultSetMetaData md = null;
        Statement stmt = null;

        try
        {
          stmt = conn.createStatement();
        //here I can get the announcement categories.
           StringBuffer myQuery = new StringBuffer();
           myQuery.append(" select d.department_name ");
          myQuery.append(" from departments d, faculty f, users u ");
          myQuery.append(" where u.user_id = f.faculty_id and ");
          myQuery.append(" d.dhead_id = f.faculty_id and ");
          myQuery.append(" u.login_id = '"+ myUser.myLoginId+"' ");
          rs= stmt.executeQuery(myQuery.toString());
          rs.next();
          myFaculty.setDepartment(rs.getString(1));
          myQuery = new StringBuffer();

          myQuery.append(" select distinct f1.faculty_id, f1.degree, f1.name, f1.last_name,f1.rank,f1.email
    ");
          myQuery.append(" from faculty f1, class_enrollment c1, courses co1 ");
          myQuery.append(" where (f1.faculty_id in ( ");
          myQuery.append(" select  f.faculty_id  ");
          myQuery.append(" from faculty f ");
          myQuery.append(" where (f.program_id in (select d3.department_no ");
          myQuery.append(" from faculty f3,departments d3,users u ");
          myQuery.append(" where (f3.faculty_id = u.user_id) and  ");
          myQuery.append(" (f3.faculty_id = d3.dhead_id) and ");
          myQuery.append(" (u.login_id = '" + myUser.myLoginId + "')))  ");
             myQuery.append(" minus ");
             myQuery.append(" select f2.faculty_id ");
```

```
          myQuery.append(" from faculty f2,users u2 ");
          myQuery.append(" where f2.faculty_id = u2.user_id and ");
          myQuery.append(" u2.login_id = '" + myUser.myLoginId +"')) ");
          myQuery.append(" and ");
          myQuery.append(" (c1.faculty_id = f1.faculty_id) ");
          myQuery.append(" and ");
          myQuery.append(" (co1.course_id = c1.course_id) ");
          rs = stmt.executeQuery(myQuery.toString());
          String degree = "";
          String name = "";
          String lname = "";
          String rank = "";
          String fId = "";
          String fId2 = "";
          String mail = "";
          int count = 1;
      while (rs.next())
        {
          fId = rs.getString(1);
          degree = rs.getString(2);
          name = rs.getString(3);
          lname = rs.getString(4);
          rank = rs.getString(5);
          mail = rs.getString(6);
          fId2 = mail.substring(0,mail.indexOf('@'));
        announcements = announcements +
                    "<tr>"+
                    "<td width=\"20\" height=\"19\" valign=\"middle\">" +
          "<INPUT onclick=\"javascript:facultyview('" + name +
                    "','" + lname + "','" + fId2 + "','" + mail + "',200,200)\" type=button value=\""+count + " -
Info" + "\">"+
          "</td>" +

          "<td width='20%' height='19'>" + degree+ "</td>"+
          "<td width='20%' height='19'>" + name + "</td>"+
          "<td width='20%' height='19'>" + lname + "</td>"+
          "<td width='20%' height='19'>"+rank+"</td>"+
         "</tr>";
        count++;

        }//end of while

        myFaculty.setAnnouncements(announcements);
        announcements = "";
        myQuery = new StringBuffer();
        myQuery.append(" select distinct f1.name, f1.last_name, ");
        myQuery.append("
c1.course_id,co1.course_name,c1.section_id,co1.course_explanation,co1.course_credit,f1.faculty_id ");
        myQuery.append(" from faculty f1, class_enrollment c1, courses co1 ");
        myQuery.append(" where (f1.faculty_id in ( ");
        myQuery.append(" select  f.faculty_id  ");
        myQuery.append(" from faculty f ");
        myQuery.append(" where (f.program_id in (select d3.department_no ");
        myQuery.append(" from faculty f3,departments d3 ");
        myQuery.append(" where (f3.last_name = 'Adem') and  ");
        myQuery.append(" (f3.faculty_id = d3.dhead_id)))  ");
```

```
        myQuery.append(" minus ");
          myQuery.append(" select f2.faculty_id ");
          myQuery.append(" from faculty f2 ");
          myQuery.append(" where f2.last_name = 'Adem') )  ");
          myQuery.append(" and ");
          myQuery.append(" (c1.faculty_id = f1.faculty_id) ");
          myQuery.append(" and  ");
          myQuery.append(" (co1.course_id = c1.course_id) ");


          rs = stmt.executeQuery(myQuery.toString());
          String courseId = "";
          String cname = "";
          String section = "";
          String instructor = "";
          String cexp = "";
          String ccredit = "";
          count = 1;

      while (rs.next())
        {
          instructor = rs.getString(1) + " " + rs.getString(2);
          courseId = rs.getString(3);
          cname = rs.getString(4);
          section = rs.getString(5);
          cexp = rs.getString(6);
          ccredit = rs.getString(7);
          fId = rs.getString(8);

        announcements = announcements +

          "<tr>"+
          "</td>" +
          "<td width=\"20\" height=\"19\" >"+
          "<INPUT onclick=\"javascript:courseview('" + courseId +
          "','" + cname + "','" + cexp + "','" + ccredit + "',400,300)\" type=button value=\""+count + " - Info" +
"\">"+
          "</td>" +
          "<td width='20%' height='19'>" +
          courseId+ "</td>"+
    "<td width='20%' height='19'>" + cname + "</td>"+
    "<td width='20%' height='19'>" +
    "<form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=">" +
    "<p style='word-spacing: 0; margin: 0'><input type='submit' value='" + "  " + section + "   " +"'
name='B1' style='font-size: 8pt'></p>"+
    " <input type='hidden' name='action' value='studentsForDepHead'>"+
    "  <input type='hidden' name='course' value='" + courseId + "'>"+
    "  <input type='hidden' name='section' value='" + section + "'>" +
    "  <input type='hidden' name='faculty_id' value='" + fId + "'>"
    + "</td>"+
    "<td width='20%' height='19'>"+instructor+"</td>"+
   "</tr>" + " </form>";
        count++;
        }//end of while
        myFaculty.setCourses(announcements);
```

```
      rs.close();
      stmt.close();
    }//end of try
    catch (Exception e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return myFaculty ;
    }

 }//end of getdepartment info

/*
Method executeCompanyInfoForStudent:executes CompanyInfo request for
student
*/
 public void executeCompanyInfoForStudent(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("company info for student requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);

    if (myUser.state) {//the result of the cookiecheck must be affirmative.

        Communicator myFaculty;
         myFaculty = new Communicator();
       //--connect to the db and get+arrange the courses for the user.

         myFaculty = getCompanyInfoForStudents(myUser);
         java.util.Date myDate = new java.util.Date();
         myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
         myFaculty.setLastName(myUser.myLastName);
         myFaculty.setOperation("companyInfo");
         myFaculty.setTime(myDate.toString());
         HttpSession session = request.getSession(true);
         session.putValue("myStudent",myFaculty);
         gotoPage("/src/webadmin/Student.jsp",request,respond);
    }else {
     logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
     }//end of if
 }//end of executeCompanyInfoForStudent.

/*
Method getCompanyInfoForStudents: gets the data from the database and sends
the data to the proper jsp file.
*/
```

```
public Communicator getCompanyInfoForStudents(UserType myUser){
  Connection conn = poolMgr.getConnection("myThesis");
  String announcements = new String();
  Communicator myStudent = new Communicator();
   boolean result = false;
  if (conn == null)
  {
      logWriter.log("database connection error",1);
    //return false;
  }
  ResultSet rs = null;
  ResultSetMetaData md = null;
  Statement stmt = null;

  StringBuffer firstInfo = new StringBuffer();
  firstInfo.append("select distinct s.student_id,s.name, ");
  firstInfo.append(" s.last_name,s.remaining_points,");
  firstInfo.append(" r.unit_name, ");
  firstInfo.append(" co.rank,co.name,co.last_name,co.command_unit ");
  firstInfo.append(" from students s, sections se, regiment r, users u,");
  firstInfo.append(" command_personel co ");
  firstInfo.append(" where s.student_id = u.user_id and ");
  firstInfo.append(" s.section_id = se.section_id and ");
  firstInfo.append(" se.company_no = r.unit_id and ");
  firstInfo.append(" co.command_unit = r.unit_id and ");
  firstInfo.append(" u.login_id = ");

  StringBuffer regInfo = new StringBuffer();
  regInfo.append("select distinct co.rank,co.name,co.last_name ");
  regInfo.append(" from  command_personel co where co.command_unit = ");

  StringBuffer awardInfo = new StringBuffer();
  awardInfo.append(" select distinct s.student_id, ");
  awardInfo.append(" a.award_no,a.award_type,a.result,a.award_date");
  awardInfo.append(" from students s, awards a, users u ");
  awardInfo.append(" where s.student_id = u.user_id and ");
  awardInfo.append(" s.student_id = a.student_id and ");
  awardInfo.append(" u.login_id = ");

  StringBuffer punishmentInfo = new StringBuffer();
  punishmentInfo.append(" select distinct s.student_id, ");
  punishmentInfo.append(" p.punishment_no,p.punishment_type, ");
  punishmentInfo.append(" p.result,p.points_taken,p.punishment_date ");
  punishmentInfo.append(" from students s, punishments p, users u ");
  punishmentInfo.append(" where s.student_id = u.user_id and ");
  punishmentInfo.append(" s.student_id = p.student_id and ");
  punishmentInfo.append(" u.login_id = ");


  try
  {
    stmt = conn.createStatement();
  //here I can get the announcement categories.
  firstInfo.append("'" + myUser.myLoginId + "'");

    rs = stmt.executeQuery(firstInfo.toString());
```

```
rs.next();


myStudent.setUserId(rs.getString(1));
myStudent.setName(rs.getString(2) + " " + rs.getString(3));
myStudent.setPoints(rs.getString(4));
myStudent.setCompanyName(rs.getString(5));
myStudent.setCompComName(rs.getString(6) + " " + rs.getString(7) +
                    " " + rs.getString(8));
int company = rs.getInt(9);
int battalion = (int) company/10;

rs = stmt.executeQuery(regInfo.toString()+"'2'");
rs.next();
myStudent.setRegComName(rs.getString(1) + " " + rs.getString(2) +
                    " " + rs.getString(3));

regInfo.append("'" + battalion + "'");
rs = stmt.executeQuery(regInfo.toString());
rs.next();
myStudent.setBatComName(rs.getString(1) + " " + rs.getString(2) +
                    " " + rs.getString(3));

awardInfo.append("'" + myUser.myLoginId + "'");
rs = stmt.executeQuery(awardInfo.toString());
String award = "";
while(rs.next()){
 rs.getString(1);
award += " <tr> "+
      "<td width=\"84\" height=\"19\">"+
      rs.getString(2) + "</td>"+
      "<td width=\"98\" height=\"19\">"+
      rs.getString(3) + "</td>"+
      "<td width=\"248\" height=\"19\">"+
      rs.getString(4) + "</td>"+
      "<td width=\"231\" height=\"19\" colspan=\"2\">"+
      rs.getString(5).substring(0,11) + "</td>"+
      " </tr>";
}
myStudent.setAwards(award);
punishmentInfo.append("'" + myUser.myLoginId + "'");
rs = stmt.executeQuery(punishmentInfo.toString());
String punishments = "";
while(rs.next()){
 rs.getString(1);
punishments += " <tr> "+
      "<td width=\"84\" height=\"19\">"+
      rs.getString(2) + "</td>"+
      "<td width=\"98\" height=\"19\">"+
      rs.getString(3) + "</td>"+
      "<td width=\"248\" height=\"19\">"+
      rs.getString(4) + "</td>"+
      "<td width=\"126\" height=\"19\"> "+
      rs.getString(5) + "</td>"+
      "<td width=\"99\" height=\"19\" >"+
      rs.getString(6).substring(0,11) + "</td>"+
```

```
            " </tr>";

      }
      myStudent.setPunishments(punishments);
      rs.close();
      stmt.close();
    }//end of try
    catch (SQLException e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return myStudent ;
    }


 }//end of get company info for student

/*
Method executeChangePassword:executes change password request for user
*/
 public void executeChangePassword(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("change password requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);

    if (myUser.state) {//the result of the cookiecheck must be affirmative.
        Communicator myFaculty;
          myFaculty = new Communicator();
        //--connect to the db and get+arrange the courses for the user.

          Object val1 = myParameterValues.get("firstText");
          Object val2 =  myParameterValues.get("secondText");

         if (val1.equals(val2)){
          setNewPassword(myUser.myLoginId,myParameterValues);

          }
          String valueName = "";
          String pageName = "";
          HttpSession session = request.getSession(true);
          if (myUser.myType.equals("FACULTY")){
             myFaculty = getProfPersonal(myUser.myLoginId);
             valueName = "myFaculty";
             pageName = "/src/webadmin/Faculty.jsp";
          }else if (myUser.myType.equals("STUDENT")){
             myFaculty = getStudentPersonal(myUser.myLoginId);
             valueName = "myStudent";
             pageName = "/src/webadmin/Student.jsp";
          }else if (myUser.myType.equals("DEP_HEAD")){
```

```
            myFaculty = getProfPersonal(myUser.myLoginId);
            valueName = "myDepHead";
            pageName = "/src/webadmin/DepartmentHead.jsp";
         }else if (myUser.myType.equals("COMMAND")){
            myFaculty = getRegimentPersonal(myUser.myLoginId);
            valueName = "myRegiment";
            pageName = "/src/webadmin/Regiment.jsp";
         }
         myFaculty.setMessage("The new passwords do not match, please try again");
         if (val1.equals(val2)){
            myFaculty.setMessage("The new passwords are OK, password is UPDATED");
         }

         java.util.Date myDate = new java.util.Date();
         myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
         myFaculty.setLastName(myUser.myLastName);
         myFaculty.setOperation("Personal_information");
         myFaculty.setTime(myDate.toString());

         session.putValue(valueName,myFaculty);
         gotoPage(pageName,request,respond);
    }else {
     logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
      }//end of if
 }//end of execute change password

/*
Method setRegimentUpdate: updates the regiment info in the database.
*/

 public void setRegimentUpdate(String myUser,Hashtable myParameterValues){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();

    if (conn == null)
    {
        logWriter.log("database connection error",1);
      //return false;
    }

    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;

    try
    {
      stmt = conn.createStatement();

    StringBuffer myQuery = new StringBuffer();
    myQuery.append("select f.person_id from command_personel f, users u ");
    myQuery.append("where u.user_id = f.person_id and u.login_id='" + myUser +"'");
    rs = stmt.executeQuery(myQuery.toString());
```

```
   rs.next();
   String person_id = rs.getString(1);
   rs.close();

   conn.setAutoCommit(false);

   StringBuffer myUpdate = new StringBuffer();
   myUpdate.append("update command_personel ");
   myUpdate.append("set email='" + myParameterValues.get("email").toString() + "'");
   myUpdate.append(" , room_no='" + myParameterValues.get("room").toString() + "'");
   myUpdate.append(" , tel_no='" + myParameterValues.get("tel").toString() + "'");
   myUpdate.append(" , address='" + myParameterValues.get("address").toString() + "'");
   myUpdate.append(" , city='" + myParameterValues.get("city").toString() + "'");
   myUpdate.append(" , province='" + myParameterValues.get("province").toString() + "'");
   myUpdate.append(" where person_id ='" + person_id + "'");
   stmt.executeUpdate(myUpdate.toString());
   conn.commit();
   conn.setAutoCommit(true);
   stmt.close();
   }//end of try
   catch (Exception e)
   {
     logWriter.log(e," Exception error",1);
   }
   finally
   {
     poolMgr.freeConnection("myThesis", conn);
   }

 }//end of set regiment personal information

/*
Method setProfUpdate: updates the faculty info in the database.
*/

 public void setProfUpdate(String myUser,Hashtable myParameterValues){
     Communicator myFaculty = new Communicator();
     Connection conn = poolMgr.getConnection("myThesis");
     String information = new String();
   if (conn == null)
   {
       logWriter.log("database connection error",1);
     //return false;
   }

   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;

   try
   {
     stmt = conn.createStatement();
   //here I can get the announcement categories.

   StringBuffer myQuery = new StringBuffer();
   myQuery.append("select f.faculty_id from faculty f, users u ");
```

```
   myQuery.append("where u.user_id = f.faculty_id and u.login_id='" + myUser +"'");
   rs = stmt.executeQuery(myQuery.toString());

   rs.next();
   String faculty_id = rs.getString(1);
   rs.close();

   StringBuffer myUpdate = new StringBuffer();
   myUpdate.append("update faculty ");
   myUpdate.append("set email='" + myParameterValues.get("email").toString() + "'");
   myUpdate.append(" , room_no='" + myParameterValues.get("room").toString() + "'");
   myUpdate.append(" , tel_no='" + myParameterValues.get("tel").toString() + "'");
   myUpdate.append(" , address='" + myParameterValues.get("address").toString() + "'");
   myUpdate.append(" , city='" + myParameterValues.get("city").toString() + "'");
   myUpdate.append(" , province='" + myParameterValues.get("province").toString() + "'");
   myUpdate.append(" where faculty_id ='" + faculty_id + "'");

   stmt.executeUpdate(myUpdate.toString());
   stmt.close();
   }//end of try
   catch (Exception e)
   {
     logWriter.log(e," Exception error",1);
   }
   finally
   {
     poolMgr.freeConnection("myThesis", conn);
   }

 }//end of set prof personal information


/*
Method setStudentUpdate: updates the student info in the database.
*/

 public void setStudentUpdate(String myUser,Hashtable myParameterValues){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();
   if (conn == null)
    {
        logWriter.log("database connection error",1);
      //return false;
   }

   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;

   try
   {
     stmt = conn.createStatement();
      //here I can get the announcement categories.
      StringBuffer myQuery = new StringBuffer();
      myQuery.append("select s.student_id from students s, users u ");
```

```
        myQuery.append("where u.user_id = s.student_id and u.login_id='" + myUser +"'");
        rs = stmt.executeQuery(myQuery.toString());
        rs.next();
        String student_id = rs.getString(1);
        conn.setAutoCommit(false);

        StringBuffer myUpdate = new StringBuffer();
        myUpdate.append("update students ");
        myUpdate.append("set email='" + myParameterValues.get("email").toString() + "'");
        myUpdate.append(" , dormitory='" + myParameterValues.get("room").toString() + "'");
        myUpdate.append(" , address='" + myParameterValues.get("address").toString() + "'");
        myUpdate.append(" , city='" + myParameterValues.get("city").toString() + "'");
        myUpdate.append(" , province='" + myParameterValues.get("province").toString() + "'");
        myUpdate.append(" where student_id ='" + student_id + "'");
       stmt.executeUpdate(myUpdate.toString());
       conn.commit();
       conn.setAutoCommit(true);
       rs.close();
       stmt.close();
     }//end of try
     catch (Exception e)
     {
       logWriter.log(e," Exception error",1);
     }
     finally
     {
       poolMgr.freeConnection("myThesis", conn);
     }

   }//end of set student personal information


/*
Method executeGradeUpdate:executes grade update request for faculty personnel
and sends respond to the proper jsp file
*/

   public void executeGradeUpdate(Hashtable myParameterValues,
                        HttpServletRequest request,
                        HttpServletResponse respond){

     UserType myUser = new UserType();
     myUser = checkCookie(request,respond);
     logWriter.log("grade update  requested by " + myUser.myLoginId +
                 " from " + request.getRemoteAddr(),2);
     if (myUser.state) {//the result of the cookiecheck must be affirmative.
        Communicator myFaculty;
        myFaculty = new Communicator();
        //--connect to the db and get+arrange the courses for the user.
        setGradeUpdate(myParameterValues,myUser);
        myFaculty = getStudentsForProf(myParameterValues,myUser);
        java.util.Date myDate = new java.util.Date();
         myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
          myFaculty.setLastName(myUser.myLastName);
          myFaculty.setOperation("StudentsForProfessor");
```

```
            myFaculty.setTime(myDate.toString());
            HttpSession session = request.getSession(true);
            session.putValue("myFaculty",myFaculty);
            gotoPage("/src/webadmin/Faculty.jsp",request,respond);
        }else {
         logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                  " at " + request.getRemoteAddr(),1);
          }//end of if
     }//end of execute grade update.


     /*
     Method setGradeUpdate: updates the grades in the database.
     */

      public Communicator setGradeUpdate(Hashtable myParameterValues,
                                  UserType myUser){
         Communicator myFaculty = new Communicator();
         Connection conn = poolMgr.getConnection("myThesis");
         Connection conn2 = poolMgr.getConnection("myThesis");
         String information = new String();
         String examStr ="";
         boolean result = false;
        if (conn == null)
        {
            logWriter.log("database connection error",1);
          //return false;
        }
        ResultSet rs = null;
        ResultSetMetaData md = null;
        Statement stmt = null;
        int examNo = 0;
        Object val = myParameterValues.get("First");
        PreparedStatement stmtUpdate = null;
        try{
          if (val != null){
            examNo = 0;
            examStr = "First";
            stmtUpdate = conn2.prepareStatement("update course_grades set first_midterm = ? where
student_id = ?");
          }else{
            val = myParameterValues.get("Second");
            if (val != null){
               examNo = 1;
               examStr = "Second";
               stmtUpdate = conn2.prepareStatement("update course_grades set second_midterm = ? where
student_id = ?");
            }else{
               val = myParameterValues.get("Final");
               if (val != null){
               examNo = 2;
               examStr = "Final";
               stmtUpdate = conn2.prepareStatement("update course_grades set final = ? where student_id = ?");
               }else{
                  val = myParameterValues.get("Grade");
                  if (val != null){
```

```
                examNo = 3;
                examStr = "Grade";
                stmtUpdate = conn2.prepareStatement("update course_grades set grade = ? where student_id =
?");
                }
              }
            }//second if
          }//first if
        }//end of try
        catch(SQLException e){
         logWriter.log(e," Exception error",1);
        }
         String course = "";
         String section = "";
         String loginId = "";
         String order = "";
         val = myParameterValues.get("course");
         course = val.toString();
         val = myParameterValues.get("section");
         section = val.toString();
         loginId = myUser.myLoginId;
         //get the course and the section name from the parameters

         myFaculty.setCourse(course);
         myFaculty.setSection(section);

         val = myParameterValues.get("order");
         String tempOrder = "";
         if (val != null) {
           if(val.equals("student_no")){
           order = "s.student_id asc";
           }else if(val.equals("name")){
           order = "s.name asc";
           }else if(val.equals("last_name")){
           order = "s.last_name asc";
           }else if(val.equals("first_midterm")){
           order = "g.first_midterm desc";
           }else if(val.equals("second_midterm")){
           order = "g.second_midterm desc";
           }else if(val.equals("final")){
           order = "g.final desc";
           }else if(val.equals("grade")){
           order = "g.course_grade asc";
           }
              tempOrder = val.toString();
         }else{
         order = "s.student_id desc";
             tempOrder = "student_no";
         }

         //---
         boolean firstExamLocked = false;//the default is not locked
         boolean secondExamLocked = false;// if they are locked you can
         boolean finalExamLocked = false;//not modify those results

         try
```

```
{
  stmt = conn.createStatement();

//here I can get the students for the prof for the specific course and section

      StringBuffer myQuery = new StringBuffer();

       myQuery.append("select distinct s.student_id,c.course_id, co.course_name,");
       myQuery.append(" s.section_id,p.program_name,s.email,s.name,");
       myQuery.append("s.last_name,g.first_midterm,g.second_midterm,");
       myQuery.append("g.final,g.course_grade");

       myQuery.append(" from class_enrollment c, faculty f, students s,");
       myQuery.append(" course_grades g, courses co, programs p, users u");

       myQuery.append(" where (f.faculty_id = c.faculty_id and");
       myQuery.append(" f.faculty_id = u.user_id and u.login_id='"+myUser.myLoginId+"'");
       myQuery.append(" and (c.section_id = s.section_id) and");
       myQuery.append(" (s.student_id = g.student_id) and (c.course_id =");
       myQuery.append(" g.course_id) and c.course_id='" + course + "' and");
       myQuery.append(" co.course_id = c.course_id and s.program_id=p.program_id");
       myQuery.append(" and s.section_id = '" + section + "')");
       myQuery.append(" order by " + order );

  //the students first
  int myId = 0;
  String myIdStr = "";
  rs = stmt.executeQuery(myQuery.toString());
 while (rs.next())
  {
  myIdStr = rs.getString(1);
  conn2.setAutoCommit(false);
  val = myParameterValues.get(myIdStr+examStr);
  stmtUpdate.setString(1,val.toString());
  stmtUpdate.setString(2,myIdStr);
  stmtUpdate.executeUpdate();

  }//end of while
  conn2.commit();
  conn2.setAutoCommit(true);

 //myFaculty.setCourses(courses);
  rs.close();
  stmt.close();
  stmtUpdate.close();
}//end of try
catch (Exception e)
{
  logWriter.log(e," Exception error",1);
}
finally
{
  poolMgr.freeConnection("myThesis", conn);
  poolMgr.freeConnection("myThesis", conn2);
  return myFaculty ;
}
```

```
  }//end of set gradesUpdate(for prof)


/*
Method executeStudentsForDepHead:executes studentsInfo request for dep head
and sends respond to the proper jsp file
*/

  public void executeStudentsForDepHead(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
     logWriter.log("students info for dephead requested by " + myUser.myLoginId +
               " from " + request.getRemoteAddr(),2);

    if (myUser.state) {//the result of the cookiecheck must be affirmative.

        Communicator myFaculty;
         myFaculty = new Communicator();
       //--connect to the db and get+arrange the courses for the user.
         myFaculty = getStudentsForDepHead(myParameterValues,myUser);
         java.util.Date myDate = new java.util.Date();
         myFaculty.setLogin(myUser.myLoginId);
          myFaculty.setName(myUser.myName);
         myFaculty.setLastName(myUser.myLastName);
         myFaculty.setOperation("StudentsForDepHead");
         myFaculty.setTime(myDate.toString());
         HttpSession session = request.getSession(true);
         session.putValue("myDepHead",myFaculty);
         gotoPage("/src/webadmin/DepartmentHead.jsp",request,respond);
    }else {
     logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
               " at " + request.getRemoteAddr(),1);
      }//end of if
  }//end of execute students for prof.

  /*
Method getStudentsForDepHead: gets the data from the database .
*/

  public Communicator getStudentsForDepHead(Hashtable myParameterValues,
                                UserType myUser){
    Communicator myFaculty = new Communicator();
    Connection conn = poolMgr.getConnection("myThesis");
    String information = new String();

    boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
     ResultSetMetaData md = null;
```

```
Statement stmt = null;



String course = "";
String section = "";
String order = "";
//get the course and the section name from the parameters

Object val = myParameterValues.get("course");
course = val.toString();
myFaculty.setCourse(course);

val = myParameterValues.get("section");
section = val.toString();
myFaculty.setSection(section);

val = myParameterValues.get("faculty_id");
String fId = val.toString();

//get the students data
        StringBuffer myQuery = new StringBuffer();
        myQuery.append("select distinct c.course_id, co.course_name,");
        myQuery.append(" s.section_id,p.program_name,s.email,s.student_id,s.name,");
        myQuery.append("s.last_name,g.first_midterm,g.second_midterm,");
        myQuery.append("g.final,g.course_grade");
        myQuery.append(" from class_enrollment c, faculty f, students s,");
        myQuery.append(" course_grades g, courses co, programs p, users u");
        myQuery.append(" where (f.faculty_id = c.faculty_id and");
        myQuery.append(" f.faculty_id = '"+ fId +"'");
        myQuery.append(" and (c.section_id = s.section_id) and");
        myQuery.append(" (s.student_id = g.student_id) and (c.course_id =");
        myQuery.append(" g.course_id) and c.course_id='" + course + "' and");
        myQuery.append(" co.course_id = c.course_id and s.program_id=p.program_id");
        myQuery.append(" and s.section_id = '" + section + "')");


 try{
   stmt = conn.createStatement();
   rs = stmt.executeQuery(myQuery.toString());

   String announcements = "";
   String temp = "";
   rs.next();
   temp = rs.getString(1);
   announcements += "<table border=\"1\" width=\"688\" height=\"149\">" +  " <tr> "+
            " <td width=\"678\" colspan=\"4\" height=\"19\" bgcolor=\"#0000FF\">" +
            " <p align=\"center\"><font color=\"#FFFF00\"><b>" +
            "THE STUDENTS TAKING THE COURSE " + course + " IN SECTION " +
            section +"</b></font></td> </tr> "+" <tr>"+
            "<td width=\"79\" height=\"1\" bgcolor=\"#C0C0C0\">"+
            "<b>Course No</b></td>"+
            "<td width=\"93\" height=\"1\" bgcolor=\"#C0C0C0\">" + course + "</td>"+
            "<td width=\"99\" height=\"1\" bgcolor=\"#C0C0C0\">"+
            "<b>Course Name</b></td>"+
            "<td width=\"147\" height=\"1\" bgcolor=\"#C0C0C0\">"+
```

```
            rs.getString(2) + "</td>"+ " </tr> " + " <tr> ";
            temp = rs.getString(3);
    announcements +=
            "<td width=\"79\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            "<b> Section No</b></td> "+
            "<td width=\"93\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            section + "</td>" +
            "<td width=\"488\" height=\"19\" bgcolor=\"#C0C0C0\" colspan=\"2\">"+
            "<b>This Section's Program is " +rs.getString(4) + "</b></td> "+
            "</tr> </table> ";
            myFaculty.setAnnouncements(announcements);
    rs = null;
    rs = stmt.executeQuery(myQuery.toString());
    String courses = "";
    String studentId = "";
    String mail="";
    while (rs.next())
    {
      mail = rs.getString(5);
      studentId = rs.getString(6);
      courses += "  <tr>"+
      "<td width=\"89\" height=\"19\" bgcolor=\"#C0C0C0\" valign=\"middle\">" +
      "<p align=\"center\">"+
      "<INPUT onclick=\"javascript:transferview('" + studentId +
      "'," + mail + ",200,200)\" type=button value=\""+studentId + "\">"+
      "</p>"+
      "</td>" +
      "<td width=\"107\" height=\"19\" bgcolor=\"#C0C0C0\">" +
      "<font size=\"2\">" + rs.getString(7) + "</font></td>"+
      "<td width=\"97\" height=\"19\" bgcolor=\"#C0C0C0\">"+
      "<font size=\"2\">" + rs.getString(8) + "</font></td>"+
      "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">"+
      "<font size=\"2\">" + rs.getString(9) + "</font></td>"+
      "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            "<font size=\"2\">" + rs.getString(10) + "</font></td>"+
      "<td width=\"55\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            "<font size=\"2\">" + rs.getString(11) + "</font></td>"+
       "<td width=\"102\" height=\"19\" bgcolor=\"#C0C0C0\">" +
             "<font size=\"2\">" + rs.getString(12) + "</font></td>"
              + " </tr> ";
    }//end of while
   myFaculty.setCourses(courses);
  rs.close();
  stmt.close();
  }//end of try
  catch (Exception e)
  {
    logWriter.log(e," Exception error",1);
  }
  finally
  {
    poolMgr.freeConnection("myThesis", conn);
    return myFaculty ;
  }

}//end of get students data
```

```
/*
Method executeStudentsForProf:executes StudentInfo request for faculty personnel
and sends respond to the proper jsp file
*/

public void executeStudentsForProf(Hashtable myParameterValues,
                    HttpServletRequest request,
                    HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);
   logWriter.log("student info for faculty requested by " + myUser.myLoginId +
              " from " + request.getRemoteAddr(),2);

   if (myUser.state) {//the result of the cookiecheck must be affirmative.

       Communicator myFaculty;
        myFaculty = new Communicator();
      //--connect to the db and get+arrange the courses for the user.

        myFaculty = getStudentsForProf(myParameterValues,myUser);
        java.util.Date myDate = new java.util.Date();
        myFaculty.setLogin(myUser.myLoginId);
        myFaculty.setName(myUser.myName);
        myFaculty.setLastName(myUser.myLastName);
        myFaculty.setOperation("StudentsForProfessor");
        myFaculty.setTime(myDate.toString());
        HttpSession session = request.getSession(true);
        session.putValue("myFaculty",myFaculty);
        gotoPage("/src/webadmin/Faculty.jsp",request,respond);
   }else {
    logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),1);
     }//end of if
}//end of execute students for prof.

/*
Method getStudentsForRegiment: gets the data from the database .
*/

 public Communicator getStudentsForProf(Hashtable myParameterValues,
                              UserType myUser){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();
   boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
```

```
String course = "";
String section = "";
String order = "";
//get the course and the section name from the parameters
Object val = myParameterValues.get("course");
course = val.toString();
myFaculty.setCourse(course);

val = myParameterValues.get("section");
section = val.toString();
myFaculty.setSection(section);
val = myParameterValues.get("order");
String tempOrder = "";
if (val != null) {
  if(val.equals("student_no")){
  order = "s.student_id asc";
  }else if(val.equals("name")){
  order = "s.name asc";
  }else if(val.equals("last_name")){
  order = "s.last_name asc";
  }else if(val.equals("first_midterm")){
  order = "g.first_midterm desc";
  }else if(val.equals("second_midterm")){
  order = "g.second_midterm desc";
  }else if(val.equals("final")){
  order = "g.final desc";
  }else if(val.equals("grade")){
  order = "g.course_grade asc";
  }
  tempOrder = val.toString();
}else{
order = "s.student_id asc";
tempOrder = "student_no";
}

//---
boolean firstExamLocked = false;//the feault is not locked
boolean secondExamLocked = false;// if they are locked you can
boolean finalExamLocked = false;//not modify those results

try
{
  stmt = conn.createStatement();
  String exam = "";
  rs = stmt.executeQuery("select locked_date, no  from locked_grades ");
  while(rs.next()){
      java.util.Date myDate1 = new java.util.Date();
      myDate1 = rs.getDate(1);

  java.util.Date myDate = new java.util.Date();
  exam=rs.getString(2);
    if (myDate.compareTo(myDate1)>0){//it means it is locked
      if (exam.equals("1")){
      firstExamLocked = true;
      }else if (exam.equals("2")){
      firstExamLocked = true;
```

```
        secondExamLocked = true;
        }else {
        firstExamLocked = true;
        secondExamLocked = true;
        finalExamLocked = true;
        }

      }
  }//end of while
   rs = null;//rs reset

//get the students for the prof for the specific course and section

        StringBuffer myQuery = new StringBuffer();

        myQuery.append("select distinct c.course_id, co.course_name,");
        myQuery.append(" s.section_id,p.program_name,s.email,s.student_id,s.name,");
        myQuery.append("s.last_name,g.first_midterm,g.second_midterm,");
        myQuery.append("g.final,g.course_grade");
        myQuery.append(" from class_enrollment c, faculty f, students s,");
        myQuery.append(" course_grades g, courses co, programs p, users u");
        myQuery.append(" where (f.faculty_id = c.faculty_id and");
        myQuery.append(" f.faculty_id = u.user_id and u.login_id='"+ myUser.myLoginId +"'");
        myQuery.append(" and (c.section_id = s.section_id) and");
        myQuery.append(" (s.student_id = g.student_id) and (c.course_id =");
        myQuery.append(" g.course_id) and c.course_id='" + course + "' and");
        myQuery.append(" co.course_id = c.course_id and s.program_id=p.program_id");
        myQuery.append(" and s.section_id = '" + section + "')");
        myQuery.append(" order by " + order );

   rs = stmt.executeQuery(myQuery.toString());
   String announcements = "";
   String temp = "";
   rs.next();
   temp = rs.getString(1);
   announcements += "<table border=\"1\" width=\"688\" height=\"149\">" +  " <tr> "+
            " <td width=\"678\" colspan=\"4\" height=\"19\" bgcolor=\"#0000FF\">" +
            " <p align=\"center\"><font color=\"#FFFF00\"><b>" +
            "THE STUDENTS TAKING THE COURSE " + course + " IN SECTION " +
            section +"</b></font></td> </tr> "+" <tr>"+
            "<td width=\"79\" height=\"1\" bgcolor=\"#C0C0C0\">"+
            "<b>Course No</b></td>"+
            "<td width=\"93\" height=\"1\" bgcolor=\"#C0C0C0\">" + course + "</td>"+
            "<td width=\"99\" height=\"1\" bgcolor=\"#C0C0C0\">"+
            "<b>Course Name</b></td>"+
            "<td width=\"147\" height=\"1\" bgcolor=\"#C0C0C0\">"+
            rs.getString(2) + "</td>"+ " </tr> " + " <tr> ";
            temp = rs.getString(3);
  announcements +=
            "<td width=\"79\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            "<b> Section No</b></td> "+
            "<td width=\"93\" height=\"19\" bgcolor=\"#C0C0C0\">"+
            section + "</td>" +
            "<td width=\"488\" height=\"19\" bgcolor=\"#C0C0C0\" colspan=\"2\">"+
            "<b>This Section's Program is " +rs.getString(4) + "</b></td> "+
            "</tr> </table> ";
```

```
        myFaculty.setAnnouncements(announcements);
rs = null;
rs = stmt.executeQuery(myQuery.toString());
String courses = "";
courses +=
 "<form method=\"POST\" "+
 "style=\"line-height: 100%; margin: 0\" "+
 "action=\"/servlet/webadmin.Academy\" onSubmit=\"\">"+
 "<table border=\"1\" width=\"689\">" +
 "<input type=\"hidden\" name=\"action\" value=\"gradeUpdate\">"+
 "<input type=\"hidden\" name=\"course\" value=\""+course+"\">"+
 "<input type=\"hidden\" name=\"section\" value=\""+section+"\">"+
 "<input type=\"hidden\" name=\"order\" value=\""+tempOrder+"\">";
String studentId = "";
String mail="";
while (rs.next())
{
mail = rs.getString(5);
studentId = rs.getString(6);
  courses += "  <tr>"+
  "<td width=\"89\" height=\"19\" bgcolor=\"#C0C0C0\" valign=\"middle\">" +
  "<p align=\"center\">"+
  "<INPUT onclick=\"javascript:transferview('" + studentId +
 "'," + mail + ",200,200)\" type=button value=\""+studentId + "\">"+
 "</p>"+
  "</td>" +
  "<td width=\"107\" height=\"19\" bgcolor=\"#C0C0C0\">" +
  "<font size=\"2\">" + rs.getString(7) + "</font></td>"+
  "<td width=\"97\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<font size=\"2\">" + rs.getString(8) + "</font></td>";

if(firstExamLocked){
  courses += "<td width=\"99\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<input readonly type=\"text\" name=\"" + studentId +"First\" size=\"5\" value=\""+
  rs.getString(9) + "\" style=\"background-color: #C0C0C0\"> </tr> ";
}else{
  courses += "<td width=\"99\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<input type=\"text\" name=\"" + studentId +"First\" size=\"5\" value=\""+
  rs.getString(9)+"\"></font></td>";
}
if(secondExamLocked){
  courses += "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<input readonly type=\"text\" name=\"" + studentId +"Second\" size=\"5\" value=\""+
  rs.getString(10) + "\" style=\"background-color: #C0C0C0\"> </tr> " ;
}else{
  courses += "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<input type=\"text\" name=\"" + studentId +"Second\" size=\"5\" value=\""+
  rs.getString(10)+"\"></font></td>";
}
if(finalExamLocked){
  courses += "<td width=\"55\" height=\"19\" bgcolor=\"#C0C0C0\">"+
  "<input readonly type=\"text\" name=\"" + studentId +"Final\" size=\"5\" value=\""+
  rs.getString(11) + "\" style=\"background-color: #C0C0C0\">"+
  "<td width=\"102\" height=\"19\" bgcolor=\"#C0C0C0\">" +
  "<input readonly type=\"text\" name=\"" + studentId +"Grade\" size=\"5\" value=\""+
  rs.getString(12) + "\" style=\"background-color: #C0C0C0\">"+
```

```
    "</td>" + " </tr> ";
  }else{
    courses += "<td width=\"55\" height=\"19\" bgcolor=\"#C0C0C0\">"+
    "<input type=\"text\" name=\"" + studentId +"Final\" size=\"5\" value=\""+
    rs.getString(11)+"\"></font></td>" +
   "<td width=\"102\" height=\"19\" bgcolor=\"#C0C0C0\">" +
    "<input type=\"text\" name=\"" + studentId +"Grade\" size=\"5\" value=\""+
    rs.getString(12)+"\"></font></td>" + " </tr> ";
  }
}//end of while

  courses += "  <tr>"+
    "<td width=\"89\" height=\"19\" bgcolor=\"#C0C0C0\" colspan=\"3\">" +
    "</td>";
  if(firstExamLocked){
    courses += "<td width=\"99\" height=\"19\" bgcolor=\"#C0C0C0\">Locked"+
    "</font></td>";
  }else{
    courses += "<td width=\"99\" height=\"19\" bgcolor=\"#C0C0C0\">"+
    "<input type=\"submit\" value=\"Update\" name=\"First\" style=\"font-size: 8pt\">"+
    "</font></td>";
  }
  if(secondExamLocked){
    courses += "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">Locked"+
    "</font></td>";
  }else{
    courses += "<td width=\"127\" height=\"19\" bgcolor=\"#C0C0C0\">"+
    "<input type=\"submit\" value=\"Update\" name=\"Second\" style=\"font-size: 8pt\">"+
    "</font></td>";
  }
  if(finalExamLocked){
    courses += "<td width=\"55\" height=\"19\" bgcolor=\"#C0C0C0\">Locked"+
    "</font></td>"+
    "<td width=\"102\" height=\"19\" bgcolor=\"#C0C0C0\">Locked" +
    "</font></td>" + " </tr> ";
  }else{
    courses += "<td width=\"55\" height=\"19\" bgcolor=\"#C0C0C0\">"+
              "<input type=\"submit\" value=\"Update\" name=\"Final\" style=\"font-size: 8pt\">"+
    "</font></td>" +
   "<td width=\"102\" height=\"19\" bgcolor=\"#C0C0C0\">" +
              "<input type=\"submit\" value=\"Update\" name=\"Grade\" style=\"font-size: 8pt\">"+
    "</font></td>" + " </tr> ";
  }
  courses += "</table>" + "</form>";
  myFaculty.setCourses(courses);
  rs.close();
  stmt.close();
}//end of try
catch (Exception e)
{
  logWriter.log(e," Exception error",1);
}
finally
{
  poolMgr.freeConnection("myThesis", conn);
  return myFaculty ;
```

```
   }

 }//end of get students data (for prof)



/*
Method executeGradesForStudent:executes grades info request for students
and sends respond to the proper jsp file
*/
public void executeGradesForStudent(Hashtable myParameterValues,
                    HttpServletRequest request,
                    HttpServletResponse respond){

   UserType myUser = new UserType();
   myUser = checkCookie(request,respond);
   logWriter.log("grades for students requested by " + myUser.myLoginId +
             " from " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.
      Communicator myFaculty;
        myFaculty = new Communicator();
      //--connect to the db and get+arrange the courses for the user.
        myFaculty = getGradesForStudent(myParameterValues,myUser);
        java.util.Date myDate = new java.util.Date();
        myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
        myFaculty.setLastName(myUser.myLastName);
        myFaculty.setOperation("gradesForStudent");
        myFaculty.setTime(myDate.toString());
        HttpSession session = request.getSession(true);
        session.putValue("myStudent",myFaculty);
        gotoPage("/src/webadmin/Student.jsp",request,respond);
   }else {
    logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
             " at " + request.getRemoteAddr(),1);
     }//end of if
 }//end of execute grades for student.

/*
Method getGradesForStudent: gets the data from the database .
*/

 public Communicator getGradesForStudent(Hashtable myParameterValues,
                          UserType myUser){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();
   boolean result = false;
   if (conn == null)
   {
      logWriter.log("database connection error",1);
   }
   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;
   String course = "";
```

```
String section = "";
String order = "";
//get the course and the section name from the parameters
Object val = myParameterValues.get("Semester");
String semester = "";
String year = "";
String semesterValue = "";
if (val != null){
  semester = val.toString();
  if (semester.equals("Year 2002 semester 1st")){
    year = "2002";
    semesterValue = "1";
  }else if (semester.equals("Year 2002 semester 2nd")){
    year = "2002";
    semesterValue = "2";
  }else if (semester.equals("Year 2001 semester 1st")){
    year = "2001";
    semesterValue = "1";
  }else if (semester.equals("Year 2001 semester 2nd")){
    year = "2001";
    semesterValue = "2";
  }else if (semester.equals("Year 2000 semester 1st")){
    year = "2000";
    semesterValue = "1";
  }else if (semester.equals("Year 2000 semester 2nd")){
    year = "2000";
    semesterValue = "2";
  }else if (semester.equals("Year 1999 semester 1st")){
    year = "1999";
    semesterValue = "1";
  } else if (semester.equals("Year 1999 semester 2nd")){
    year = "1999";
    semesterValue = "2";
  }
} else {
  semester= "Year 2002 semester 2nd";
  year = "2002";
  semesterValue = "2";

}
myFaculty.setSemester(semester);
try
{
   stmt = conn.createStatement();
   String exam = "";
//here I can get the grades for the student for the specific semester
        StringBuffer myQuery = new StringBuffer();
        myQuery.append("select s.student_id,s.name,s.last_name from students s,users u ");
        myQuery.append(" where u.user_id = s.student_id and ");
        myQuery.append(" u.login_id ='" + myUser.myLoginId + "'");
        rs = stmt.executeQuery(myQuery.toString());
        rs.next();
        String sId = rs.getString(1);
        String StudentName = rs.getString(2);
        String StudentLastName = rs.getString(3);
        myFaculty.setName(StudentName);
```

```
      myFaculty.setLastName(StudentLastName);
      myFaculty.setUserId(sId);
myQuery = new StringBuffer();
 //-------------
myQuery.append(" select distinct f.name,f.last_name, c.course_id, ");
myQuery.append(" co.course_name,g.first_midterm, ");
myQuery.append(" g.second_midterm,g.final,g.course_grade, ");
myQuery.append(" f.email,co.credits,co.course_explanation ");
myQuery.append(" from class_enrollment c, faculty f, ");
myQuery.append(" students s, course_grades g,users u, ");
myQuery.append(" courses co ");
myQuery.append(" where u.user_id = s.student_id and ");
myQuery.append(" s.student_id = g.student_id and ");
myQuery.append(" s.section_id = c.section_id and ");
myQuery.append(" c.course_id = g.course_id and ");
myQuery.append(" c.faculty_id = f.faculty_id and ");
myQuery.append(" u.login_id = '" + myUser.myLoginId + "' and ");
myQuery.append(" c.course_id = co.course_id and ");
myQuery.append(" g.year_taken = '" + year + "' and ");
myQuery.append(" g.semestre_taken = '" + semesterValue + "'");
      rs = stmt.executeQuery(myQuery.toString());
String fname = "";
String fLname = "";
String cId = "";
String cName = "";
String firstExam = "";
String secondExam = "";
String finalExam = "";
String courseGrade = "";
String fmail = "";
String courseExp = "";
String courseCredit = "";
String fId = "";
String grades = "";

 while (rs.next())
 {
   fname = rs.getString(1);
   fLname =rs.getString(2);
   cId = rs.getString(3);
   cName = rs.getString(4);
   firstExam = rs.getString(5);
   secondExam = rs.getString(6);
   finalExam = rs.getString(7);
   courseGrade = rs.getString(8);
   fmail = rs.getString(9);
   fId =  fname.substring(0,1).concat(fLname);
   courseExp = rs.getString(10);
   courseCredit = rs.getString(11);
   String profName = fname + " " + fLname;
   for(int i=(11-profName.length());i>=0;i--){
      profName += " ";
   }
   grades += "<tr>"+
   "<td width=\"86\" height=\"22\" bgcolor=\"#C0C0C0\" valign=\"middle\">" +
   "<p align=\"center\">"+
```

```
          "<INPUT onclick=\"javascript:facultyview('" + fname +
         "','" + fLname + "','" + fId + "','" + fmail + "',200,200)\" type=button value=\""+profName + "\">"+
         "</p>"+
          "</td>" +
          "<td width=\"108\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<p align=\"center\">"+
          "<INPUT onclick=\"javascript:courseview('" + cId +
         "','" + cName + "','" + courseExp + "','" + courseCredit + "',400,300)\" type=button value=\""+cId +
"\">"+
         "</p>"+
          "</td>" +
          "<td width=\"94\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<font size=\"2\">"+cName +"</font></td>"+
          "<td width=\"106\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<font size=\"2\">" + firstExam + "</font></td>"+
          "<td width=\"125\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<font size=\"2\">" + secondExam + "</font></td>"+
          "<td width=\"63\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<font size=\"2\">" + finalExam + "</font></td>"+
          "<td width=\"94\" height=\"22\" bgcolor=\"#C0C0C0\">"+
          "<font size=\"2\">" + courseGrade + "</font></td>"+
             "  </tr>";
       }//end of while
     myFaculty.setCourses(grades);
     rs.close();
     stmt.close();
    }//end of try
    catch (Exception e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return myFaculty ;
    }

  }//end of get grades data (for student)


 /*
 Method executeUpdate:executes update request for users
 and sends respond to the proper jsp file
 */

  public void executeUpdate(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("information update  requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);

    if (myUser.state) {//the result of the cookiecheck must be affirmative.
       Communicator myFaculty;
```

```
        myFaculty = new Communicator();
        //--connect to the db and get+arrange the courses for the user.
        java.util.Date myDate = new java.util.Date();
        if (myUser.myType.equals("FACULTY")){
            setProfUpdate(myUser.myLoginId,myParameterValues);
            myFaculty = getProfPersonal(myUser.myLoginId);
            myFaculty.setMessage("The personal information updated");
            myFaculty.setLogin(myUser.myLoginId);
            myFaculty.setTime(myDate.toString());
            myFaculty.setOperation("Personal_information");
            myFaculty.setMessage("To change the password use the bottom form");
            HttpSession session = request.getSession(true);
            session.putValue("myFaculty",myFaculty);
            gotoPage("/src/webadmin/Faculty.jsp",request,respond);
        }else if (myUser.myType.equals("DEP_HEAD")){
            setProfUpdate(myUser.myLoginId,myParameterValues);
            myFaculty = getProfPersonal(myUser.myLoginId);
            myFaculty.setMessage("The personal information updated");
            myFaculty.setLogin(myUser.myLoginId);
            myFaculty.setTime(myDate.toString());
            myFaculty.setOperation("Personal_information");
            myFaculty.setMessage("To change the password use the bottom form");
            HttpSession session = request.getSession(true);
            session.putValue("myDepHead",myFaculty);
            gotoPage("/src/webadmin/DepartmentHead.jsp",request,respond);
        }else if (myUser.myType.equals("COMMAND")){
            setRegimentUpdate(myUser.myLoginId,myParameterValues);
            myFaculty = getRegimentPersonal(myUser.myLoginId);
            myFaculty.setMessage("The personal information updated");
            myFaculty.setLogin(myUser.myLoginId);
            myFaculty.setTime(myDate.toString());
            myFaculty.setOperation("Personal_information");
            myFaculty.setMessage("To change the password use the bottom form");
            HttpSession session = request.getSession(true);
            session.putValue("myRegiment",myFaculty);
            gotoPage("/src/webadmin/Regiment.jsp",request,respond);
         }else{//student
            setStudentUpdate(myUser.myLoginId,myParameterValues);
            myFaculty = getStudentPersonal(myUser.myLoginId);
            myFaculty.setMessage("The personal information updated");
            myFaculty.setLogin(myUser.myLoginId);
            myFaculty.setTime(myDate.toString());
            myFaculty.setOperation("Personal_information");
            myFaculty.setMessage("To change the password use the bottom form");
            HttpSession session = request.getSession(true);
            session.putValue("myStudent",myFaculty);
            gotoPage("/src/webadmin/Student.jsp",request,respond);
        }
    }else {
      logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),1);
    }//end of if
  }//end of execute update

  /*
Method setNewPassword: updates the new password in the database
```

```
*/

 public void setNewPassword(String myUser,Hashtable myParameterValues){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();
   if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;

    try
    {
      stmt = conn.createStatement();
     //here I can get the announcement categories.
     Object val1 = myParameterValues.get("firstText");
     StringBuffer myQuery = new StringBuffer();
     myQuery.append("update users set password = '" + val1.toString());
     myQuery.append("' where login_id='" + myUser +"'");
     conn.setAutoCommit(false);
     stmt.executeUpdate(myQuery.toString());
     conn.commit();
     conn.setAutoCommit(true);
      rs.close();
      stmt.close();
    }//end of try
    catch (Exception e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
    }

 }//end of get prof personal information



/*
Method executePersonal:executes personal info request for user
and sends respond to the proper jsp file
*/
 public void executePersonal(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("personal info  requested by " + myUser.myLoginId +
              " from " + request.getRemoteAddr(),2);
    if (myUser.state) {//the result of the cookiecheck must be affirmative.
        Communicator myFaculty;
```

```
        myFaculty = new Communicator();
      //--connect to the db and get+arrange the courses for the user.
        java.util.Date myDate = new java.util.Date();
        if (myUser.myType.equals("FACULTY")){
           myFaculty = getProfPersonal(myUser.myLoginId);
           myFaculty.setLogin(myUser.myLoginId);
           myFaculty.setTime(myDate.toString());
           myFaculty.setOperation("Personal_information");
           myFaculty.setMessage("To change the password use the bottom form");
           HttpSession session = request.getSession(true);
           session.putValue("myFaculty",myFaculty);
           gotoPage("/src/webadmin/Faculty.jsp",request,respond);
        }else if (myUser.myType.equals("COMMAND")){
           myFaculty = getRegimentPersonal(myUser.myLoginId);
           myFaculty.setName(myUser.myName);
           myFaculty.setLastName(myUser.myLastName);
           myFaculty.setLogin(myUser.myLoginId);
           myFaculty.setTime(myDate.toString());
           myFaculty.setOperation("Personal_information");
           myFaculty.setMessage("To change the password use the bottom form");
           HttpSession session = request.getSession(true);
           session.putValue("myRegiment",myFaculty);
           gotoPage("/src/webadmin/Regiment.jsp",request,respond);
        }else if (myUser.myType.equals("DEP_HEAD")){
           myFaculty = getProfPersonal(myUser.myLoginId);
           myFaculty.setLogin(myUser.myLoginId);
           myFaculty.setTime(myDate.toString());
           myFaculty.setOperation("Personal_information");
           myFaculty.setMessage("To change the password use the bottom form");
           HttpSession session = request.getSession(true);
           session.putValue("myDepHead",myFaculty);
           gotoPage("/src/webadmin/DepartmentHead.jsp",request,respond);
        }else {
           myFaculty = getStudentPersonal(myUser.myLoginId);
           myFaculty.setLogin(myUser.myLoginId);
           myFaculty.setTime(myDate.toString());
           myFaculty.setOperation("Personal_information");
           myFaculty.setMessage("To change the password use the bottom form");
           HttpSession session = request.getSession(true);
           session.putValue("myStudent",myFaculty);
           gotoPage("/src/webadmin/Student.jsp",request,respond);
        }
     }else {
       logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
              " at " + request.getRemoteAddr(),1);
     }//end of if
  }//end of execute personal information.

/*
Method getRegimentPersonal: gets the data from the database .
*/

  public Communicator getRegimentPersonal(String myUser){
     Communicator myFaculty = new Communicator();
     Connection conn = poolMgr.getConnection("myThesis");
     String information = new String();
```

```
 boolean result = false;
if (conn == null)
{
    logWriter.log("database connection error",1);
}
ResultSet rs = null;
ResultSetMetaData md = null;
Statement stmt = null;
try
{
  stmt = conn.createStatement();

    StringBuffer myQuery = new StringBuffer();

      myQuery.append("SELECT distinct f.name,f.mid_name,f.last_name,");
      myQuery.append("r.unit_name,f.rank,f.email,f.room_no,");
      myQuery.append("f.tel_no,f.address,f.city,f.province");
      myQuery.append(" from users u, command_personel f, regiment r ");
      myQuery.append("where f.command_unit = r.unit_id and ");
      myQuery.append("f.person_id = u.user_id and u.login_id='"+myUser+"'");

  rs = stmt.executeQuery(myQuery.toString());

  rs.next();// no while because there is only one result

    myFaculty.setLogin1(myUser);
    myFaculty.setName1(rs.getString(1));
    myFaculty.setMidName1(rs.getString(2));
    myFaculty.setLastName1(rs.getString(3));
    myFaculty.setCompanyName(rs.getString(4));
    myFaculty.setRank(rs.getString(5));
    myFaculty.setEmail(rs.getString(6));
    myFaculty.setRoom(rs.getString(7));
    myFaculty.setTel(rs.getString(8));
    myFaculty.setAddress(rs.getString(9));
    myFaculty.setCity(rs.getString(10));
    myFaculty.setProvince(rs.getString(11));


  rs.close();
  stmt.close();
}//end of try
catch (Exception e)
{
 logWriter.log(e," Exception error",1);
}
finally
{
  poolMgr.freeConnection("myThesis", conn);
  return myFaculty ;
}

}//end of get command personal information

/*
Method getProfPersonal: gets the data from the database .
```

```
*/

 public Communicator getProfPersonal(String myUser){
    Communicator myFaculty = new Communicator();
    Connection conn = poolMgr.getConnection("myThesis");
    String information = new String();
    boolean result = false;
   if (conn == null)
   {
       logWriter.log("database connection error",1);
   }
   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;

   try
   {
     stmt = conn.createStatement();
        StringBuffer myQuery = new StringBuffer();

           myQuery.append("SELECT distinct f.name,f.mid_name,f.last_name,");
           myQuery.append("f.degree,d.department_name,f.rank,f.email,f.room_no,");
           myQuery.append("f.tel_no,f.address,f.city,f.province");
           myQuery.append(" from users u, faculty f, departments d ");
           myQuery.append("where f.program_id = d.department_no and ");
           myQuery.append("f.faculty_id = u.user_id and u.login_id='"+myUser+"'");

     rs = stmt.executeQuery(myQuery.toString());
     rs.next();// no while because there is only one result

        myFaculty.setName(rs.getString(1));
        myFaculty.setMidName(rs.getString(2));
        myFaculty.setLastName(rs.getString(3));
        myFaculty.setDegree(rs.getString(4));
        myFaculty.setDepartment(rs.getString(5));
        myFaculty.setRank(rs.getString(6));
        myFaculty.setEmail(rs.getString(7));
        myFaculty.setRoom(rs.getString(8));
        myFaculty.setTel(rs.getString(9));
        myFaculty.setAddress(rs.getString(10));
        myFaculty.setCity(rs.getString(11));
        myFaculty.setProvince(rs.getString(12));


     rs.close();
     stmt.close();
   }//end of try
   catch (Exception e)
   {
     logWriter.log(e," Exception error",1);
   }
   finally
   {
     poolMgr.freeConnection("myThesis", conn);
     return myFaculty ;
   }
```

```
 }//end of get prof personal information

/*
Method getStudentPersonal: gets the data from the database .
*/

  public Communicator getStudentPersonal(String myUser){
   Communicator myFaculty = new Communicator();
   Connection conn = poolMgr.getConnection("myThesis");
   String information = new String();
   boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;

    try
    {
      stmt = conn.createStatement();
        StringBuffer myQuery = new StringBuffer();

           myQuery.append("SELECT distinct s.name,s.mid_name,s.last_name,");
           myQuery.append("s.section_id,p.program_name,re.unit_name,s.email,s.dormitory,");
           myQuery.append("do.location,s.address,s.city,s.province, s.student_id ");
           myQuery.append(" from users u, students s, programs p, dormitory do,sections se ,");
           myQuery.append(" regiment re ");
           myQuery.append("where s.program_id = p.program_id and ");
           myQuery.append("s.dormitory = do.dormitory_no and s.section_id = se.section_id and ");
           myQuery.append("se.company_no = re.unit_id and ");
           myQuery.append("s.student_id = u.user_id and u.login_id='"+myUser+"'");



      rs = stmt.executeQuery(myQuery.toString());

      rs.next();// no while because there is only one result

        myFaculty.setName(rs.getString(1));
        myFaculty.setMidName(rs.getString(2));
        myFaculty.setLastName(rs.getString(3));
        myFaculty.setSection(rs.getString(4));
        myFaculty.setDepartment(rs.getString(5));
        myFaculty.setRegiment(rs.getString(6));
        myFaculty.setEmail(rs.getString(7));
        myFaculty.setRoom(rs.getString(8));
        myFaculty.setLocation(rs.getString(9));
        myFaculty.setAddress(rs.getString(10));
        myFaculty.setCity(rs.getString(11));
        myFaculty.setProvince(rs.getString(12));
        myFaculty.setUserId(rs.getString(13));

      rs.close();
```

```
      stmt.close();
    }//end of try
    catch (Exception e)
    {
     logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return myFaculty ;
    }

 }//end of get student personal information


/*
Method executeHomePage:executes home page request for user
and sends respond to the proper jsp file
*/
 public void executeHomePage(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("home page requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);
    if (myUser.state) {//the result of the cookiecheck must be affirmative.
        if (myUser.myType.equals("FACULTY")){
            handleFaculty(myUser,request,respond);
         }else if (myUser.myType.equals("STUDENT")){
            handleStudent(myUser,request,respond);
         }else if (myUser.myType.equals("COMMAND")){
            handleRegiment(myUser,request,respond);
         }else if (myUser.myType.equals("DEP_HEAD")){
            handleDepHead(myUser,request,respond);
         }
      }else {
       logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
    }//end of if
 }//end of execute prof home page.


/*
Method executeSchedule:executes schedule info for student and faculty personnel
and sends respond to the proper jsp file
*/


 public void executeSchedule(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){

    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
```

```
   logWriter.log("Schedule info requested by " + myUser.myLoginId +
            " from " + request.getRemoteAddr(),2);
   if (myUser.state) {//the result of the cookiecheck must be affirmative.
      Communicator myFaculty, myStudent;
      myFaculty = new Communicator();
      myStudent = new Communicator();
      //--connect to the db and get+arrange the courses for the user.
       myFaculty.setSchedule(getSchedule(myUser));
       java.util.Date myDate = new java.util.Date();
       myFaculty.setLogin(myUser.myLoginId);
       myFaculty.setName(myUser.myName);
       myFaculty.setLastName(myUser.myLastName);
       myFaculty.setOperation("prof_schedule");
       myFaculty.setTime(myDate.toString());
       if (myUser.myType.equals("FACULTY")){
           HttpSession session = request.getSession(true);
           session.putValue("myFaculty",myFaculty);
           gotoPage("/src/webadmin/Faculty.jsp",request,respond);
       }else{
           myStudent = myFaculty;
           HttpSession session = request.getSession(true);
           session.putValue("myStudent",myStudent);
           gotoPage("/src/webadmin/Student.jsp",request,respond);
       }
   }else {
     logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
            " at " + request.getRemoteAddr(),1);
   }//end of if
 }//end of execute professor schedule.
/*
Method getSchedule: gets the data from the database .
*/

 public String getSchedule(UserType myUser){
    Connection conn = poolMgr.getConnection("myThesis");
    String schedule = new String();
    String courseId = new String();
    String sectionId = new String();
    String classRoom = new String();
    String courseName = new String();
    int classTime,classDate;
    boolean isStudent = false;
    boolean result = false;

   if (conn == null)
   {
      logWriter.log("database connection error",1);
   }
   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;

   try
   {
     stmt = conn.createStatement();
   //here I can get the announcement categories.
```

```
CourseType myCourses[][];
myCourses = new CourseType[7][7];
CourseType thisCourse = new CourseType();
//initialize the array;
for (int x = 1 ; x  < 7 ; x++){
  for (int y = 1 ; y < 7 ; y++){
     myCourses[x][y] = new CourseType();
  }
}

StringBuffer myQuery = new StringBuffer();
if (myUser.myType.equals("FACULTY")){
   myQuery.append("SELECT distinct c.course_id, c.section_id ,");
   myQuery.append(" c.classroom_no, c.course_time, course_date,");
   myQuery.append(" co.course_name from class_enrollment c, users u,");
   myQuery.append(" courses co where c.faculty_id = u.user_id and");
   myQuery.append(" u.login_id='" + myUser.myLoginId);
   myQuery.append("' and c.course_id = co.course_id ");

}else{
   myQuery.append("SELECT distinct c.course_id, f.last_name ,");
   myQuery.append(" c.classroom_no, c.course_time, course_date,");
   myQuery.append(" f.name from class_enrollment c, users u,");
   myQuery.append(" students s, faculty f,");
   myQuery.append(" courses co where s.student_id = u.user_id and");
   myQuery.append(" u.login_id='" + myUser.myLoginId);
   myQuery.append("' and c.course_id = co.course_id ");
   myQuery.append(" and c.section_id = s.section_id ");
   myQuery.append(" and c.faculty_id = f.faculty_id ");
   isStudent = true;
  }

rs = stmt.executeQuery(myQuery.toString());
while (rs.next())
{
  courseId = rs.getString(1);
  sectionId = rs.getString(2);//for students it means professor
  classRoom = rs.getString(3);
  classTime = Integer.valueOf(rs.getString(4)).intValue();
  classDate = Integer.valueOf(rs.getString(5)).intValue();
  courseName = rs.getString(6);
  myCourses[classTime][classDate].courseId = courseId;
  myCourses[classTime][classDate].sectionId = sectionId;
  myCourses[classTime][classDate].classRoom = classRoom;
  myCourses[classTime][classDate].courseName = courseName;


}//end of while
String myTime[] = {"0830","0930","1030","1130","1330","1430"};
for (int x = 1 ; x  <= 6 ; x++){
schedule += " <tr>  <td width=\"8%\" height=\"20\"> <p style=\"word-spacing: 0;" +
        " margin-top: 0; margin-bottom: 0\">"+
        "<p align=\"center\" style=\"word-spacing: 0; margin-top: 0; margin-bottom: 0\">"+
        "<font size=\"1\">" + x +
        "</font></td> <td width=\"8%\" height=\"20\">"+
```

```
            "<p align=\"center\" style=\"word-spacing: 0; margin-top: 0; margin-bottom: 0\">"+
            "<font size=\"1\">" + myTime[x-1] +
            "</font></td>";
          for (int y = 1 ; y  <= 5 ; y++){
             if (myCourses[x][y].courseId.equals(" ")){

        schedule += " <td width=\"16%\" height=\"20\">" +
        "<p style=\"word-spacing: 0; margin: 0\"><font size=\"1\"> </font></p>"+
        "<p style=\"word-spacing: 0; margin: 0\"><font size=\"1\"> </font></p>"+
        "<p style=\"word-spacing: 0; margin: 0\"><font size=\"1\"> </font></td>";

             } else {
        schedule += "<td width=\"16%\" height=\"20\" bgcolor=\"#00FF00\">"+
        "<p style=\"word-spacing: 0; margin: 0\"><font size=\"1\">"+
        "Course: "+ myCourses[x][y].courseId +
        "</font></p> <p style=\"word-spacing: 0; margin: 0\"><font size=\"1\">"+
        "Room:  " + myCourses[x][y].classRoom +
        "</font></p>  <p style=\"word-spacing: 0; margin: 0\"><font size=\"1\">";
             if (isStudent){
        schedule += "Professor:" +  myCourses[x][y].sectionId + "," +
             myCourses[x][y].courseName + " </font></td> ";
             }else{
       schedule += "Section:" +  myCourses[x][y].sectionId +
             "</font></td> ";
             }
            }//end if
          }//end of first for
          schedule +="</tr>";
          } //end of second for


      rs.close();
      stmt.close();

    }//end of try
    catch (Exception e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return schedule;
    }

  }//end of get schedule.


/*
Method executeCourses:executes course request for student and faculty personnel
and sends respond to the proper jsp file
*/

  public void executeCourses(Hashtable myParameterValues,
                      HttpServletRequest request,
                      HttpServletResponse respond){
```

```
    UserType myUser = new UserType();
    myUser = checkCookie(request,respond);
    logWriter.log("Course info  requested by " + myUser.myLoginId +
                " from " + request.getRemoteAddr(),2);
    if (myUser.state) {//the result of the cookiecheck must be affirmative.
         Communicator myFaculty, myStudent;
         myFaculty = new Communicator();
         myStudent = new Communicator();
       //--connect to the db and get+arrange the courses for the user.

         java.util.Date myDate = new java.util.Date();
         myFaculty.setLogin(myUser.myLoginId);
         myFaculty.setName(myUser.myName);
         myFaculty.setLastName(myUser.myLastName);
         myFaculty.setOperation("courses");
         myFaculty.setTime(myDate.toString());
         if (myUser.myType.equals("FACULTY")){
            myFaculty.setCourses(getProfCourses(myUser));
            HttpSession session = request.getSession(true);
            session.putValue("myFaculty",myFaculty);
            gotoPage("/src/webadmin/Faculty.jsp",request,respond);
         }else{
            myFaculty.setCourses(getStudentCourses(myUser));
            myStudent = myFaculty;
            HttpSession session = request.getSession(true);
            session.putValue("myStudent",myStudent);
            gotoPage("/src/webadmin/Student.jsp",request,respond);
         }

    }else {
       logWriter.log("cookie check is in valid for  " + myUser.myLoginId +
                " at " + request.getRemoteAddr(),1);
    }//end of if
  }//end of execute courses.

/*
  Method getProfCourses: gets the data from the database .
  */

  public String getProfCourses(UserType myUser){
    Connection conn = poolMgr.getConnection("myThesis");
    String announcements = new String();
    boolean result = false;
     if (conn == null)
     {
         logWriter.log("database connection error",1);
     }
     ResultSet rs = null;
     ResultSetMetaData md = null;
     Statement stmt = null;

     try
     {
       stmt = conn.createStatement();
```

```
    StringBuffer myQuery = new StringBuffer();

       myQuery.append("SELECT distinct c.course_id, co.course_name,");
       myQuery.append("co.course_explanation, co.credits, c.section_id  from class_enrollment c,");
       myQuery.append(" users u, courses co ");
       myQuery.append(" where c.faculty_id = u.user_id");
       myQuery.append(" and u.login_id='" + myUser.myLoginId + "' and c.course_id = co.course_id ");

    rs = stmt.executeQuery(myQuery.toString());

    while (rs.next())
    {
      String courseId = rs.getString(1);
      String courseName = rs.getString(2);
      String courseExp = rs.getString(3);
      String courseCred = rs.getString(4);
      String sectionId = rs.getString(5);
      announcements = announcements +
              "<tr> <td width=\"8%\" bgcolor=\"#C0C0C0\" valign=\"middle\">"+
              "<p align=\"center\">"+
              "<form method=\"POST\""+
              "style=\"line-height: 100%; margin: 0\""+
              " action=\"" +
              "/servlet/webadmin.Academy\""+
              "onSubmit=\"\" style=\"text-align: Center; text-indent: 0; margin: 0\">"+
              "<p style =\"line-height: 100%; margin: 0\">"+
              "<input type=\"Submit\" value=\"Show\" name=\"B1\"></p>"+
              "<input type=\"hidden\" name=\"action\" value=\"studentsForProf\">"+
              "<input type=\"hidden\" name=\"course\" value=\""+ courseId +
              "\"><input type=\"hidden\" name=\"section\" value=\"" + sectionId +"\">"+
              " </form>"+
              "</td>"+
              "<td width=\"9%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseId+ "</td>"+
              "<td width=\"11%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseName+ "</td>"+
              "<td width=\"40%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseExp+ "</td>"+
              "<td width=\"13%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseCred+"</td>"+
              "<td width=\"9%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+sectionId+"</td>"+
              "</tr>";
    }//end of while
    rs.close();
    stmt.close();
  }//end of try
  catch (SQLException e)
  {
    logWriter.log(e," Exception error",1);
  }
  finally
  {
    poolMgr.freeConnection("myThesis", conn);
    return announcements ;
  }

 }//end of getcourses
/*
Method getStudentCourses: gets the data from the database .
*/
```

```
  public String getStudentCourses(UserType myUser){
    Connection conn = poolMgr.getConnection("myThesis");
    String courses = new String();
    boolean result = false;
     if (conn == null)
     {
         logWriter.log("database connection error",1);
     }
     ResultSet rs = null;
     ResultSetMetaData md = null;
     Statement stmt = null;

     try
     {
       stmt = conn.createStatement();

       StringBuffer myQuery = new StringBuffer();
          myQuery.append("SELECT distinct c.course_id, co.course_name,");
          myQuery.append("co.course_explanation, co.credits, f.name , f.last_name  from class_enrollment
c,");
          myQuery.append(" users u, courses co , faculty f , students s");
          myQuery.append(" where s.student_id = u.user_id and s.section_id = c.section_id ");
          myQuery.append(" and c.faculty_id = f.faculty_id ");
          myQuery.append(" and u.login_id='" + myUser.myLoginId + "' and c.course_id = co.course_id ");
       rs = stmt.executeQuery(myQuery.toString());
       int count = 0;//# of courses the student is taking.
       while (rs.next())
       {
         count++;
         String courseId = rs.getString(1);
         String courseName = rs.getString(2);
         String courseExp = rs.getString(3);
         String courseCred = rs.getString(4);
         String profName = rs.getString(5);
         String profLastName = rs.getString(6);
       courses = courses +
                  "<tr>"+
                  "<td width=\"9%\" bgcolor=\"#C0C0C0\" valign=\"middle\"> <font size=\"2\">"+count+
"</td>"+
                  "<td width=\"9%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseId+ "</td>"+
                  "<td width=\"11%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseName+ "</td>"+
                  "<td width=\"40%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseExp+ "</td>"+
                  "<td width=\"13%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+courseCred+"</td>"+
                  "<td width=\"9%\" bgcolor=\"#C0C0C0\"> <font size=\"2\">"+profName+" "
+profLastName +"</td>"+
                  "</tr>";

       }//end of while

       rs.close();
       stmt.close();
     }//end of try
     catch (SQLException e)
     {
       logWriter.log(e," Exception error",1);
```

```
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return courses ;
    }
 }//end of getStudentCourses

/*
Method executeLogin:executes first login for user
and calls the the proper handler method
*/

 public void executeLogin(Hashtable myParameterValues,
                          HttpServletRequest request,
                          HttpServletResponse respond){

   UserType myUser = new UserType();
   //read time
   java.util.Date myTime = new java.util.Date();
   String myStrTime = "" + myTime.getTime();
   //get the login id and the password
   Object val = myParameterValues.get("loginId");
   if (val != null) {
      myUser.myLoginId = val.toString();
   } else {
      logWriter.log(" loginId is not entered",1);
   }
   val = myParameterValues.get("password");
   if (val != null) {
      myUser.myPassword = val.toString();
   } else {
     logWriter.log(" password not entered",1);
   }
   //check the database if the loginid+password are correct.

   if(checkLoginId(myUser)) {//login is OK.record the login,create cookie,give the usercode
     String host = request.getRemoteAddr().toString();
     recordLogin(myUser.myLoginId,myStrTime,host);
     //create and save a cookie with the login id and time.
     Cookie cookie = new Cookie("loginId", myUser.myLoginId);
     respond.addCookie(cookie);
     cookie = new Cookie("time",myStrTime );
     respond.addCookie(cookie);

     cookie = new Cookie ("name", myUser.myName);
     respond.addCookie(cookie);
     cookie = new Cookie ("lastName", myUser.myLastName);
     respond.addCookie(cookie);
     cookie = new Cookie ("userCode", myUser.myType);
     respond.addCookie(cookie);

     //get the userCode professor/student/department heads/commander
       if (myUser.myType.equals("FACULTY")){
         //goto execute faculty function
         handleFaculty(myUser,request,respond);
```

```
        } else if (myUser.myType.equals("COMMAND")){
          handleRegiment(myUser,request,respond);
        } else if (myUser.myType.equals("DEP_HEAD")){
          handleDepHead(myUser,request,respond);
        } else if (myUser.myType.equals("STUDENT")){
         handleStudent(myUser,request,respond);
        }

   }
   else {//login is false resend the login page
//   resendLogin(respond);
    logWriter.log("sending the login page one more time to " + request.getRemoteAddr(),2);
   try {
    handleReLogin("Your loginId or password is wrong, please try again",request,respond);
   //gotoPage("/src/webadmin/Login.jsp",request,respond);
   }catch (Exception e){

   }
   }
   }//end of executeLogin
/*
  Method handleDepHead: gets the announcements for the DepHead.
  */

   public void handleDepHead(UserType myUser,
                        HttpServletRequest request,
                        HttpServletResponse respond){
     Communicator myStudent;
     myStudent = new Communicator();
    //--connect to the db and get+arrange the announcements for the user.
     myStudent.setAnnouncements(getAnnouncements(myUser));
     java.util.Date myDate = new java.util.Date();
     myStudent.setLogin(myUser.myLoginId);
     myStudent.setName(myUser.myName);
     myStudent.setLastName(myUser.myLastName);
     myStudent.setOperation("announcements");
     myStudent.setTime(myDate.toString());
     HttpSession session = request.getSession(true);
     session.putValue("myDepHead",myStudent);
     gotoPage("/src/webadmin/DepartmentHead.jsp",request,respond);

   }//end of handleDepHead

   /*
   Method handleRegiment: gets the announcements for the Regiment.
   */

   public void handleRegiment(UserType myUser,
                        HttpServletRequest request,
                        HttpServletResponse respond){
     Communicator myStudent;
     myStudent = new Communicator();
    //--connect to the db and get+arrange the announcements for the user.
     myStudent.setAnnouncements(getAnnouncements(myUser));
     java.util.Date myDate = new java.util.Date();
     myStudent.setLogin(myUser.myLoginId);
```

```
    myStudent.setName(myUser.myName);
    myStudent.setLastName(myUser.myLastName);
    myStudent.setOperation("announcements");
    myStudent.setTime(myDate.toString());
    HttpSession session = request.getSession(true);
    session.putValue("myRegiment",myStudent);
    gotoPage("/src/webadmin/Regiment.jsp",request,respond);

 }//end of handleRegiment
/*
 Method handleStudent: gets the announcements for the Student.
 */
  public void handleStudent(UserType myUser,
                        HttpServletRequest request,
                        HttpServletResponse respond){
    Communicator myStudent;
    myStudent = new Communicator();
   //--connect to the db and get+arrange the announcements for the user.
    myStudent.setAnnouncements(getAnnouncements(myUser));
    java.util.Date myDate = new java.util.Date();
    myStudent.setLogin(myUser.myLoginId);
    myStudent.setName(myUser.myName);
    myStudent.setLastName(myUser.myLastName);
    myStudent.setOperation("announcements");
    myStudent.setTime(myDate.toString());
    HttpSession session = request.getSession(true);
    session.putValue("myStudent",myStudent);
    gotoPage("/src/webadmin/Student.jsp",request,respond);

 }//end of handleStudent

/*
 Method handleFaculty: gets the announcements for the faculty.
 */

  public void handleFaculty(UserType myUser,
                        HttpServletRequest request,
                        HttpServletResponse respond){
    Communicator myFaculty;
    myFaculty = new Communicator();
    //--connect to the db and get+arrange the announcements for the user.
    myFaculty.setAnnouncements(getAnnouncements(myUser));
    java.util.Date myDate = new java.util.Date();
    myFaculty.setLogin(myUser.myLoginId);
    myFaculty.setName(myUser.myName);
    myFaculty.setLastName(myUser.myLastName);
    myFaculty.setOperation("announcements");
    myFaculty.setTime(myDate.toString());
    HttpSession session = request.getSession(true);
    session.putValue("myFaculty",myFaculty);
    gotoPage("/src/webadmin/Faculty.jsp",request,respond);

 }//end of handleDaculty

/*
 Method getAnnouncements: gets the announcements from the database.
```

```
*/
public String getAnnouncements(UserType myUser){

  Connection conn = poolMgr.getConnection("myThesis");
  String announcements = new String();
  boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
    //get the announcements");

    try
    {
      stmt = conn.createStatement();
    //get the announcement categories.
      StringBuffer myQuery = new StringBuffer();
      myQuery.append( "SELECT announcement, category, pub_date from announcements where ");

    if (myUser.myType.equals("FACULTY")){
      myQuery.append(" (category = 3) or (category = 6) or (category = 1)");
    }else if (myUser.myType.equals("COMMAND")){
      myQuery.append(" (category = 1) or (category = 4) or (category = 7)");
    } else if (myUser.myType.equals("STUDENT")){
       myQuery.append(" (category = 1) or (category = 5) or (category = 7) or (category = 6) ");
    } else {
       myQuery.append(" (category = 1) or (category = 2)");
    }

      rs = stmt.executeQuery(myQuery.toString());
      int count = 1;

      while (rs.next())
      {
      announcements = announcements +
                "<tr> <td width=\"15%\" bgcolor=\"#C0C0C0\">" + count + "</td>" +
                "<td width=\"51%\" bgcolor=\"#C0C0C0\">" + rs.getString("announcement")
                + "</td>" + "<td width=\"34%\" bgcolor=\"#C0C0C0\">"
                + rs.getString("pub_date") + "</td> </tr>";
      count++;
      }//end of while

      rs.close();
      stmt.close();
    }//end of try
    catch (SQLException e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return announcements ;
```

```
   }
 }//end of getannouncements

 /*
Method chackLoginId:checks if the login is correct or not
*/

 public boolean checkLoginId(UserType myUser){

 Connection conn = poolMgr.getConnection("myThesis");
 boolean result = false;
   if (conn == null)
   {
       logWriter.log("database connection error",1);
   }
   ResultSet rs = null;
   ResultSetMetaData md = null;
   Statement stmt = null;
   try
   {
     stmt = conn.createStatement();
     StringBuffer myQuery = new StringBuffer();
     myQuery.append("SELECT LOGIN_ID,PASSWORD,USER_CODE,user_id");
     myQuery.append(" FROM USERS WHERE LOGIN_ID ='" + myUser.myLoginId +"'");

     rs = stmt.executeQuery(myQuery.toString());

     while (rs.next())
     {
       String newLogin = rs.getString("LOGIN_ID");
       String newPassword = rs.getString("PASSWORD");
       if (newLogin.equals(myUser.myLoginId) && (newPassword.equals(myUser.myPassword))){
          myUser.myType = rs.getString("USER_CODE");
          myUser.myUserId = rs.getString("user_id");
          result = true;
       }//end of if
     }//end of while


    if (!result) {
       logWriter.log("login has problems",1);
    }else{
      myQuery = new StringBuffer();
      myQuery.append("select name,last_name from ");
      if ((myUser.myType.equals("FACULTY")) || (myUser.myType.equals("DEP_HEAD"))){
         myQuery.append("faculty where faculty_id ='"+myUser.myUserId+"'");
      }else if ((myUser.myType.equals("STUDENT"))){
         myQuery.append("students where student_id ='"+myUser.myUserId+"'");
      }else if ((myUser.myType.equals("COMMAND"))){
         myQuery.append("command_personel where person_id ='"+myUser.myUserId+"'");
      }
      rs = null;
      rs = stmt.executeQuery(myQuery.toString());
      while (rs.next()){
         myUser.myName = rs.getString(1);
         myUser.myLastName = rs.getString(2);
```

```
        }

      }
    rs.close();
    stmt.close();
    }//end of try
    catch (SQLException e)
    {
      logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return result;
    }
 }//end of checkLoginId

 /*
Method recordLogin:records the login and login time in to the database
*/

 public boolean recordLogin(String loginId, String myStrTime, String host){

   Connection conn = poolMgr.getConnection("myThesis");
   boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    //ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
    try
    {
      stmt = conn.createStatement();
      StringBuffer myQuery = new StringBuffer();
      myQuery.append("insert into login_records values ('");
      myQuery.append(loginId + "','" + myStrTime+ "','" + host + "')");
      stmt.executeUpdate(myQuery.toString());
      //rs.close();
      stmt.close();
    }//end of try
    catch (SQLException e)
    {
     logWriter.log(e," Exception error",1);
    }
    finally
    {
      poolMgr.freeConnection("myThesis", conn);
      return result;
    }
 }//end of recordLogin

 /*
Method checkCookie:checks if the current request and the cookies are the same.
*/
```

```
private UserType checkCookie(HttpServletRequest request,
                HttpServletResponse response) {
  Cookie[] myCookie = request.getCookies();
  UserType myUser = new UserType();
  //String user = "";
  String userTime = "";
  for(int i=0; i<myCookie.length; i++) {
      Cookie cookie = myCookie[i];
      if (cookie.getName().equals("loginId")){
        myUser.myLoginId = cookie.getValue();
      } else if (cookie.getName().equals("time")){
        userTime = cookie.getValue();
      } else if (cookie.getName().equals("name")){
        myUser.myName = cookie.getValue();
      } else if (cookie.getName().equals("lastName")){
        myUser.myLastName = cookie.getValue();
      } else if (cookie.getName().equals("userCode")){
        myUser.myType = cookie.getValue();
      }

    }//end of for
    if (checkLoginIdTime(myUser.myLoginId,userTime,request,response)){
      myUser.state = true;
    }else {
      myUser.state = false;
    }
      return myUser;
 }//end of checkcookie

 /*
Method checkLoginIdTime:checks if the cookie is valid/same with the database
*/

 private boolean checkLoginIdTime(String user, String userTime,
                HttpServletRequest request,
                HttpServletResponse response){

     Connection conn = poolMgr.getConnection("myThesis");
     boolean result = false;
    if (conn == null)
    {
        logWriter.log("database connection error",1);
    }
    ResultSet rs = null;
    ResultSetMetaData md = null;
    Statement stmt = null;
    try
    {
      stmt = conn.createStatement();
      StringBuffer myQuery = new StringBuffer();
      myQuery.append("SELECT LOGIN_ID,LOGIN_TIME ");
      myQuery.append("FROM LOGIN_RECORDS ");
      rs = stmt.executeQuery(myQuery.toString());
      while (rs.next())
      {
```

```
        String newLogin = rs.getString("LOGIN_ID");
        String newTime = rs.getString("LOGIN_TIME");
        if (newLogin.equals(user) && (newTime.equals(userTime))){
        result = true;
        }//end of if
      }//end of while
      rs.close();
      stmt.close();
      poolMgr.freeConnection("myThesis", conn);
      if (result){
      return result;//it means everything is ok.
      } else {
     //login name and the time is not matching. it means fraud.
     //now I will close the connections and send the login page one more time
      //illegal access;
      handleReLogin("Access denied, Please login again", request, response);
      //send the login page write the login bean a message.
     }//end of if
     return false;
   }//end of try
   catch (SQLException e)
   {
     logWriter.log(e," Exception error",1);
   }
   finally
   {

     return result;
   }
 }//end of checkLoginIdTime

 /*
Method handleReLogin:send the login page to the user
*/

 public void handleReLogin(String message,
                     HttpServletRequest request,
                     HttpServletResponse respond){
   Communicator myLogin;
   myLogin = new Communicator();
   myLogin.setMessage(message);
   HttpSession session = request.getSession(true);
   session.putValue("myLogin",myLogin);
   gotoPage("/src/webadmin/Login.jsp",request,respond);

}//end of handleReLogin

 /*
Method gotoPage:this method used to send the proper jsp files as respond.
*/

private void gotoPage(String address,
                HttpServletRequest request,
                HttpServletResponse response)
  {
  try {
```

```
  RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher(address);
  dispatcher.forward(request, response);
  }
  catch (Exception e){
     logWriter.log(e," Exception error",1);
  }

 }//end of gotopage

 /*
 class UserType:has the attributes of the users.
 */

 class UserType{
 String myLoginId = "Rasim";
 String myPassword = "";
 String myType= "";
 String myUserId = "";
 String myName ="";
 String myLastName ="";
 boolean state = false;//this is used for result.
 }

  /*
 Class CourseType:has the attributes of the courses
 */

 class CourseType{
   String courseName=" ";
   String courseId=" ";
   String sectionId=" ";
   String classRoom=" ";
   String classTime=" ";
   String classDate=" ";
   }

}//end of the class Academy servlet
```

2. Source Code of the Connection Pool Class

```
/**
 * Title: ConnectionPool.java
 * Description: Creates a connection pool. Logs the activities
 * @version 1.0
 */

package webadmin;

import java.sql.*;
import java.util.*;
import java.io.*;

public class ConnectionPool
{
```

```
private String name;
private String URL;
private String user;
private String password;
private int maxConns;
private int timeOut;
private LogWriter logWriter;


private int checkedOut;
private Vector freeConnections = new Vector();

public ConnectionPool(String name, String URL, String user,
  String password, int maxConns, int initConns, int timeOut,
  PrintWriter pw, int logLevel)
  {

  this.name = name;
  this.URL = URL;
  this.user = user;
  this.password = password;
  this.maxConns = maxConns;
  this.timeOut = timeOut > 0 ? timeOut : 5;

  logWriter = new LogWriter(name, logLevel, pw);
  initPool(initConns);

  logWriter.log("New pool created", LogWriter.INFO);
  String lf = System.getProperty("line.separator");
  logWriter.log(lf +
            " url=" + URL + lf +
            " user=" + user + lf +
            " password=" + password + lf +
            " initconns=" + initConns + lf +
            " maxconns=" + maxConns + lf +
            " logintimeout=" + this.timeOut, LogWriter.DEBUG);
  logWriter.log(getStats(), LogWriter.DEBUG);
}

private void initPool(int initConns)
{
  for (int i = 0; i < initConns; i++)
  {
    try
    {
      Connection pc = newConnection();
      freeConnections.addElement(pc);
    }
    catch (SQLException e)
    { }
  }
}

public Connection getConnection() throws SQLException
{
  logWriter.log("Request for connection received", LogWriter.DEBUG);
```

```
  try
  {
    return getConnection(timeOut * 1000);
  }
   catch (SQLException e)
  {
    logWriter.log(e, "Exception getting connection",
              LogWriter.ERROR);
    throw e;
  }
}

private synchronized Connection getConnection(long timeout)
              throws SQLException
{

  // Get a pooled Connection from the cache or a new one.
  // Wait if all are checked out and the max limit has
  // been reached.
  long startTime = System.currentTimeMillis();
  long remaining = timeout;
  Connection conn = null;
  while ((conn = getPooledConnection()) == null)
  {
    try
    {
      logWriter.log("Waiting for connection. Timeout=" + remaining,
                LogWriter.DEBUG);
      wait(remaining);
    }
    catch (InterruptedException e)
    { }
    remaining = timeout - (System.currentTimeMillis() - startTime);
    if (remaining <= 0)
    {
      // Timeout has expired
      logWriter.log("Time-out while waiting for connection",
                LogWriter.DEBUG);
      throw new SQLException("getConnection() timed-out");
    }
  }

  // Check if the Connection is still OK
  if (!isConnectionOK(conn))
  {
    // It was bad. Try again with the remaining timeout
    logWriter.log("Removed selected bad connection from pool",
              LogWriter.ERROR);
    return getConnection(remaining);
  }
  checkedOut++;
  logWriter.log("Delivered connection from pool", LogWriter.INFO);
  logWriter.log(getStats(), LogWriter.DEBUG);
  return conn;
}
```

```
private boolean isConnectionOK(Connection conn)
{
  Statement testStmt = null;
  try
  {
    if (!conn.isClosed())
    {
      // Try to createStatement to see if it's really alive
      testStmt = conn.createStatement();
      testStmt.close();
    }
    else
    {
      return false;
    }
  }
  catch (SQLException e)
  {
    if (testStmt != null)
    {
      try
      {
        testStmt.close();
      }
      catch (SQLException se)
      { }
    }
    logWriter.log(e, "Pooled Connection was not okay",
                  LogWriter.ERROR);
    return false;
  }
  return true;
}

private Connection getPooledConnection() throws SQLException
{
  Connection conn = null;
  if (freeConnections.size() > 0)
  {
    // Pick the first Connection in the Vector
    // to get round-robin usage
    conn = (Connection) freeConnections.firstElement();
    freeConnections.removeElementAt(0);
  }
  else if (maxConns == 0 || checkedOut < maxConns)
  {
    conn = newConnection();
  }
  return conn;
}

private Connection newConnection() throws SQLException
{
  Connection conn = null;
  if (user == null) {
    conn = DriverManager.getConnection(URL);
```

```
    }
    else {
      conn = DriverManager.getConnection(URL, user, password);
    }
    logWriter.log("Opened a new connection", LogWriter.INFO);
    return conn;
  }

  public synchronized void freeConnection(Connection conn)
  {
    // Put the connection at the end of the Vector
    freeConnections.addElement(conn);
    checkedOut--;
    notifyAll();
    logWriter.log("Returned connection to pool", LogWriter.INFO);
    logWriter.log(getStats(), LogWriter.DEBUG);
  }

  public synchronized void release()
  {
    Enumeration allConnections = freeConnections.elements();
    while (allConnections.hasMoreElements())
    {
      Connection con = (Connection) allConnections.nextElement();
      try
      {
        con.close();
        logWriter.log("Closed connection", LogWriter.INFO);
      }
      catch (SQLException e)
      {
        logWriter.log(e, "Couldn't close connection", LogWriter.ERROR);
      }
    }
    freeConnections.removeAllElements();
  }

  private String getStats() {
    return "Total connections: " +
      (freeConnections.size() + checkedOut) +
      " Available: " + freeConnections.size() +
      " Checked-out: " + checkedOut;
  }


}//end of connectionPool
```

3. Source Code of the Connection Pool Manager Class

```
/**
 * Title: PoolManager.java
 * Description: USed to manage the Connection pool
 * @version 1.0
 */

package webadmin;
```

```
import java.sql.*;
import java.io.*;
import java.util.*;

public class PoolManager
{

  static private PoolManager instance;
  static private int clients;

  private LogWriter logWriter;
  private PrintWriter pw;

  private Vector drivers = new Vector();
  private Hashtable pools = new Hashtable();

  private PoolManager()
  {
    init();
  }


  static synchronized public PoolManager getInstance()
  {
    if (instance == null)
    {
      instance = new PoolManager();
    }
    clients++;
    return instance;
  }

  private void init()
  {
    // Log to System.err until we have read the logfile property
    pw = new PrintWriter(System.err, true);
    logWriter = new LogWriter("PoolManager", LogWriter.INFO, pw);
    InputStream is = getClass().getResourceAsStream("database.properties");
    Properties dbProps = new Properties();
    try
    {
      dbProps.load(is);
    }
    catch (Exception e)
    {
      logWriter.log("Can't read the properties file. " +
        "Make sure database.properties is in the CLASSPATH",
        LogWriter.ERROR);
      return;
    }
    String logFile = dbProps.getProperty("logfile");
    if (logFile != null)
    {
      try
      {
        pw = new PrintWriter(new FileWriter(logFile, true), true);
```

```
        logWriter.setPrintWriter(pw);
      }
      catch (IOException e)
      {
        logWriter.log("Can't open the log file: " + logFile +
          ". Using System.err instead", LogWriter.ERROR);
      }
    }
    loadDrivers(dbProps);
    createPools(dbProps);
  }

  private void loadDrivers(Properties props)
  {
    String driverClasses = props.getProperty("drivers");
    StringTokenizer st = new StringTokenizer(driverClasses);
    while (st.hasMoreElements())
    {
      String driverClassName = st.nextToken().trim();
      try
      {
        Driver driver = (Driver)
          Class.forName(driverClassName).newInstance();
        DriverManager.registerDriver(driver);
        drivers.addElement(driver);
        logWriter.log("Registered JDBC driver " + driverClassName,
          LogWriter.INFO);
      }
      catch (Exception e)
      {
        logWriter.log(e, "Can't register JDBC driver: " +
          driverClassName, LogWriter.ERROR);
      }
    }
  }

  private void createPools(Properties props)
  {
    Enumeration propNames = props.propertyNames();
    while (propNames.hasMoreElements())
    {
      String name = (String) propNames.nextElement();
      if (name.endsWith(".url"))
      {
        String poolName = name.substring(0, name.lastIndexOf("."));
        String url = props.getProperty(poolName + ".url");
        if (url == null)
        {
          logWriter.log("No URL specified for " + poolName,
            LogWriter.ERROR);
          continue;
        }

        String user = props.getProperty(poolName + ".user");
        String password = props.getProperty(poolName + ".password");
```

```
String maxConns = props.getProperty(poolName +
  ".maxconns", "0");
int max;
try
{
  max = Integer.valueOf(maxConns).intValue();
}
catch (NumberFormatException e)
{
  logWriter.log("Invalid maxconns value " + maxConns +
              " for " + poolName, LogWriter.ERROR);
  max = 0;
}

String initConns = props.getProperty(poolName +
              ".initconns", "0");
int init;
try
{
  init = Integer.valueOf(initConns).intValue();
}
catch (NumberFormatException e)
{
  logWriter.log("Invalid initconns value " + initConns +
              " for " + poolName, LogWriter.ERROR);
  init = 0;
}

String loginTimeOut = props.getProperty(poolName +
  ".logintimeout", "5");
int timeOut;
try
{
  timeOut = Integer.valueOf(loginTimeOut).intValue();
}
catch (NumberFormatException e)
{
  logWriter.log("Invalid logintimeout value " + loginTimeOut +
              " for " + poolName, LogWriter.ERROR);
  timeOut = 5;
}

String logLevelProp = props.getProperty(poolName +
            ".loglevel", String.valueOf(LogWriter.ERROR));
int logLevel = LogWriter.INFO;
if (logLevelProp.equalsIgnoreCase("none"))
{
  logLevel = LogWriter.NONE;
}
else if (logLevelProp.equalsIgnoreCase("error"))
{
  logLevel = LogWriter.ERROR;
}
else if (logLevelProp.equalsIgnoreCase("debug"))
{
  logLevel = LogWriter.DEBUG;
```

```
        }

        ConnectionPool pool =
                new ConnectionPool(poolName, url, user, password,
                        max, init, timeOut, pw, logLevel);
        pools.put(poolName, pool);
      }
    }
  }

  public Connection getConnection(String name)
  {
    Connection conn = null;
    ConnectionPool pool = (ConnectionPool) pools.get(name);
    if (pool != null)
    {
      try
      {
        conn = pool.getConnection();
      }
      catch (SQLException e)
      {
        logWriter.log(e, "Exception getting connection from " +
          name, LogWriter.ERROR);
      }
    }
    return conn;
  }

  public void freeConnection(String name, Connection con)
  {
    ConnectionPool pool = (ConnectionPool) pools.get(name);
    if (pool != null)
    {
      pool.freeConnection(con);
    }
  }

  public synchronized void release()
  {
    // Wait until called by the last client
    if (--clients != 0)
    {
      return;
    }

    Enumeration allPools = pools.elements();
    while (allPools.hasMoreElements())
    {
      ConnectionPool pool = (ConnectionPool) allPools.nextElement();
      pool.release();
    }

    Enumeration allDrivers = drivers.elements();
    while (allDrivers.hasMoreElements())
    {
```

```
        Driver driver = (Driver) allDrivers.nextElement();
        try
        {
          DriverManager.deregisterDriver(driver);
          logWriter.log("Deregistered JDBC driver " +
                    driver.getClass().getName(), LogWriter.INFO);
        }
        catch (SQLException e)
        {
          logWriter.log(e, "Couldn't deregister JDBC driver: " +
                     driver.getClass().getName(), LogWriter.ERROR);
        }
      }
    }
}//end of connection pool manager
```

4. Source Code of the Communicator Java Bean Class

```
/**
 * Title: Communicator.java
 * Description: The special bean used for communicating between servlet and jps files
 * @author : Rasim Topuz, Ltjg, Turkish Navy
 * @version 1.0
 */

package webadmin;

public class Communicator {
  private String login         = "";
  private String login1         = "";
  private String time          = "";
  private String announcements  = "";
  private String operation      = "";
  private String courses        = "";
  private String schedule       = "";
  private String name           = "";
  private String lastName       = "";
  private String midName        = "";
  private String name1          = "";
  private String lastName1      = "";
  private String midName1       = "";
  private String degree         = "";
  private String department     = "";
  private String rank           = "";
  private String email          = "";
  private String room           = "";
  private String tel            = "";
  private String address        = "";
  private String city           = "";
  private String province       = "";
  private String course        ="";
  private String section       ="";
  private String message        ="";
  private String location      ="";
  private String regiment       ="";
  private String userId        ="";
```

```
private String semester      ="";
private String points        ="";
private String companyName    ="";
private String regComName     ="";
private String batComName     ="";
private String compComName    ="";
private String awards        ="";
private String punishments    ="";

    /**Access operation property*/
public String getPunishments() {
  return punishments;
}
/**Access sample property*/
public void setPunishments(String newValue) {
  if (newValue!=null) {
    punishments = newValue;
  }
}
    /**Access operation property*/
public String getAwards() {
  return awards;
}
/**Access sample property*/
public void setAwards(String newValue) {
  if (newValue!=null) {
    awards = newValue;
  }
}
    /**Access operation property*/
public String getCompComName() {
  return compComName;
}
/**Access sample property*/
public void setCompComName(String newValue) {
  if (newValue!=null) {
    compComName = newValue;
  }
}

    /**Access operation property*/
public String getBatComName() {
  return batComName;
}
/**Access sample property*/
public void setBatComName(String newValue) {
  if (newValue!=null) {
    batComName = newValue;
  }
}
    /**Access operation property*/
public String getRegComName() {
  return regComName;
}
/**Access sample property*/
public void setRegComName(String newValue) {
```

```
  if (newValue!=null) {
    regComName = newValue;
  }
}
    /**Access operation property*/
public String getCompanyName() {
  return companyName;
}
/**Access sample property*/
public void setCompanyName(String newValue) {
  if (newValue!=null) {
    companyName = newValue;
  }
}
    /**Access operation property*/
public String getPoints() {
  return points;
}
/**Access sample property*/
public void setPoints(String newValue) {
  if (newValue!=null) {
    points = newValue;
  }
}
    /**Access operation property*/
public String getSemester() {
  return semester;
}
/**Access sample property*/
public void setSemester(String newValue) {
  if (newValue!=null) {
    semester = newValue;
  }
}


    /**Access operation property*/
public String getUserId() {
  return userId;
}
/**Access sample property*/
public void setUserId(String newValue) {
  if (newValue!=null) {
    userId = newValue;
  }
}
    /**Access operation property*/
public String getRegiment() {
  return regiment;
}
/**Access sample property*/
public void setRegiment(String newValue) {
  if (newValue!=null) {
    regiment = newValue;
  }
}
```

```
   /**Access operation property*/
public String getLocation() {
  return location;
}
/**Access sample property*/
public void setLocation(String newValue) {
  if (newValue!=null) {
    location = newValue;
  }
}
 /**Access operation property*/
public String getMessage() {
  return message;
}
/**Access sample property*/
public void setMessage(String newValue) {
  if (newValue!=null) {
    message = newValue;
  }
}
   /**Access operation property*/
public String getSection() {
  return section;
}
/**Access sample property*/
public void setSection(String newValue) {
  if (newValue!=null) {
    section = newValue;
  }
}
     /**Access operation property*/
public String getCourse() {
  return course;
}
/**Access sample property*/
public void setCourse(String newValue) {
  if (newValue!=null) {
    course = newValue;
  }
}

  /**Access operation property*/
public String getProvince() {
  return province;
}
/**Access sample property*/
public void setProvince(String newValue) {
  if (newValue!=null) {
    province = newValue;
  }
}
  /**Access operation property*/
public String getCity() {
  return city;
}
/**Access sample property*/
```

```
  public void setCity(String newValue) {
    if (newValue!=null) {
      city = newValue;
    }
  }
    /**Access operation property*/
  public String getAddress() {
    return address;
  }

  /**Access sample property*/
  public void setAddress(String newValue) {
    if (newValue!=null) {
       address = newValue;
//      address.concat(newValue);
       //address = new String();
//      address = new String("                                    ");
//      for (int i=0;i<=newValue.length()-1;i++){
  //    address += newValue.charAt(i);
     //  }
       //      System.out.println(address);


     //  address =  new String(newValue);
    }
  }
    /**Access operation property*/
  public String getTel() {
    return tel;
  }
  /**Access sample property*/
  public void setTel(String newValue) {
    if (newValue!=null) {
      tel = newValue;
    }
  }
    /**Access operation property*/
  public String getRoom() {
    return room;
  }
  /**Access sample property*/
  public void setRoom(String newValue) {
    if (newValue!=null) {
      room = newValue;
    }
  }
    /**Access operation property*/
  public String getRank() {
    return rank;
  }
  /**Access sample property*/
  public void setRank(String newValue) {
    if (newValue!=null) {
      rank = newValue;
    }
  }
```

```
  /**Access operation property*/
public String getEmail() {
  return email;
}
/**Access sample property*/
public void setEmail(String newValue) {
  if (newValue!=null) {
    email = newValue;
  }
}
  /**Access operation property*/
public String getDepartment() {
  return department;
}
/**Access sample property*/
public void setDepartment(String newValue) {
  if (newValue!=null) {
    department = newValue;
  }
}
  /**Access operation property*/
public String getDegree() {
  return degree;
}
/**Access sample property*/
public void setDegree(String newValue) {
  if (newValue!=null) {
    degree = newValue;
  }
}
  /**Access operation property*/
public String getMidName() {
  return midName;
}
/**Access sample property*/
public void setMidName(String newValue) {
  if (newValue!=null) {
    midName = newValue;
  }
}
  /**Access operation property*/
public String getLastName() {
  return lastName;
}
/**Access sample property*/
public void setLastName(String newValue) {
  if (newValue!=null) {
    lastName = newValue;
  }
}

//-----
 /**Access operation property*/
public String getMidName1() {
  return midName1;
```

```
}
/**Access sample property*/
public void setMidName1(String newValue) {
  if (newValue!=null) {
    midName1 = newValue;
  }
}
  /**Access operation property*/
public String getLastName1() {
  return lastName1;
}
/**Access sample property*/
public void setLastName1(String newValue) {
  if (newValue!=null) {
    lastName1 = newValue;
  }
}
  /**Access operation property*/
public String getLogin() {
  return login;
}
/**Access sample property*/
public void setLogin(String newValue) {
  if (newValue!=null) {
    login = newValue;
  }
}
/**Access operation property*/
public String getLogin1() {
  return login1;
}
/**Access sample property*/
public void setLogin1(String newValue) {
  if (newValue!=null) {
    login1 = newValue;
  }
}
/**Access operation property*/
public String getSchedule() {
  return schedule;
}
/**Access sample property*/
public void setSchedule(String newValue) {
  if (newValue!=null) {
    schedule = newValue;
  }
}
  /**Access operation property*/
public String getCourses() {
  return courses;
}
/**Access sample property*/
public void setCourses(String newValue) {
  if (newValue!=null) {
    courses = newValue;
  }
```

```
  }
  /**Access operation property*/
  public String getOperation() {
    return operation;
  }
  /**Access sample property*/
  public void setOperation(String newValue) {
    if (newValue!=null) {
      operation = newValue;
    }
  }

  /**Access sample property*/
  public String getName() {
    return name;
  }
  /**Access sample property*/
  public void setName(String newValue) {
    if (newValue!=null) {
      name = newValue;
    }
  }


  /**Access sample property*/
  public String getName1() {
    return name1;
  }
  /**Access sample property*/
  public void setName1(String newValue) {
    if (newValue!=null) {
      name1 = newValue;
    }
  }
    /**Access functionType property*/
  public String getTime() {
    return time;
  }
  /**Access functionType property*/
  public void setTime(String newValue) {
    if (newValue!=null) {
      time = newValue;
    }
  }

  public String getAnnouncements() {
    return announcements;
  }
  /**Access functionType property*/
  public void setAnnouncements(String newValue) {
    if (newValue!=null) {
      announcements = newValue;
    }
  }
}// end of communicator Java bean
```

5. Source Code of the Log Writer Class

```
/**
 * Title: LogWriter.java
 * Description: records the log activities.
 * @version 1.0
 */

package webadmin;

import java.io.*;
import java.util.*;

public class LogWriter
{
  public static final int NONE = 0;
  public static final int ERROR = 1;
  public static final int INFO = 2;
  public static final int DEBUG = 3;

  private static final String ERROR_TEXT = "error";
  private static final String INFO_TEXT = "info";
  private static final String DEBUG_TEXT = "debug";

  private PrintWriter pw;
  private String owner;
  private int logLevel;

  public LogWriter(String owner, int logLevel, PrintWriter pw)
  {
    this.pw = pw;
    this.owner = owner;
    this.logLevel = logLevel;
  }

  public LogWriter(String owner, int logLevel)
  {
    this(owner, logLevel, null);
  }

  public int getLogLevel()
  {
    return logLevel;
  }

  public PrintWriter getPrintWriter()
  {
    return pw;
  }

  public void setLogLevel(int logLevel)
  {
    this.logLevel = logLevel;
  }

  public void setPrintWriter(PrintWriter pw)
```

```
  {
    this.pw = pw;
  }

  public void log(String msg, int severityLevel)
  {
    if (pw != null)
    {
      if (severityLevel <= logLevel)
      {
        pw.println("[" + new Date() + "]  " +
        getSeverityString(severityLevel) + ": " +
        owner + ": " + msg);
      }
    }
  }

  public void log(Throwable t, String msg, int severityLevel)
  {
    log(msg + " : " + toTrace(t), severityLevel);
  }

  private String getSeverityString(int severityLevel)
  {
    switch (severityLevel)
    {
      case ERROR:
        return ERROR_TEXT;
      case INFO:
        return INFO_TEXT;
      case DEBUG:
        return DEBUG_TEXT;
      default:
        return "Unknown";
    }
  }

  private String toTrace(Throwable e)
  {
    StringWriter sw = new StringWriter();
    PrintWriter pw = new PrintWriter(sw);
    e.printStackTrace(pw);
    pw.flush();
    return sw.toString();
  }
}
```

6. Source Code of the DepartmentHead JSP File

```
<!--File name: Faculty.jsp
   Purposes : To handle the Faculty users requests.-->

 <%@ page errorPage="ErrorPage.jsp" %>
<HTML>
```

```
<HEAD>
<jsp:useBean id="myDepHead" scope="session" class="webadmin.Communicator" />
<jsp:setProperty name="myDepHead" property="*" />
<TITLE>
Department Head
</TITLE>
<base target="_self">
<meta name="Microsoft Border" content="none">
</head>

<body bgcolor="#C8C8C8">
<SCRIPT language=JavaScript>

var width,height
var image,ext,studentId
var cond1,cond2
function facultyview(name,lastName,login,mail,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Faculty info</TITLE>"
      var s11="<CENTER>";
    var s2 ="<IMG SRC='/src/webadmin/" + login + ".jpg' BORDER=0>"
    var s14=" </CENTER><p><CENTER><font size='2'>" + name +
        " " + lastName + "</font></a>"

    var s15=" </CENTER><p><CENTER><a href='mailto:"+ mail+ "'><font size='2'>" +
        "send mail to " + login + "</font></a>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s2+s14+s15+s3+s4)
    ImageWindow.document.close()
}
var width,height
var image,ext,studentId
var cond1,cond2
function transferview(studentId,mail,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Student Picture</TITLE>"
      var s11="<CENTER>";
    var s15=" </CENTER><p><CENTER><a href='mailto:"+ mail+ "'><font size='2'>" +
        "send mail to " + studentId + "</font></a><p> </p>"

    var s2 ="<IMG SRC='/src/webadmin/" + studentId + ".jpg' BORDER=0>"
```

```
    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s2+s15+s3+s4)
    ImageWindow.document.close()
}
var width,height
var cId,cName,cCredit,ext,studentId
var cond1,cond2
function courseview(cId,cName,cExp,Credit,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Course info</TITLE>"
      var s11="<CENTER>";

      var s12 = "<table border='1' width='74%' height='106'> <tr>"
      var s13 = "<td width='41%' height='47'><b>Course id      : </b>"+
cId + "</td>"
      var s14 = "<td width='59%' height='141' rowspan='3'><b>Course Explanation : </b>" +cExp + "</td>
</tr>  <tr>"
      var s15 = "<td width='41%' height='47'><b>Course credit: </b>" + cCredit + "</td>  </tr>  <tr>"
      var s16 = "<td width='41%' height='47'><b>Course name: </b>" + cName + "</td> </tr></table>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s12+s13+s14+s15+s16+s3+s4)
    ImageWindow.document.close()
}
</SCRIPT>

<table border="0" width="100%" cellspacing="1" height="0">
  <tr>
    <td width="100%" colspan="2" valign="bottom">
    <p        align="center"><img        border="0"        src="/src/webadmin/academy.gif"        width="770"
height="100"></p></td>

  </tr>
  <tr>
    <td width="100%" colspan="2" valign="bottom"> 
      <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
       <jsp:getProperty name="myDepHead" property="name" />
         <jsp:getProperty name="myDepHead" property="lastName" />
        </p>
      <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
        <jsp:getProperty name="myDepHead" property="time" />
       </p>
```

```
   <p style="text-indent: 0; word-spacing: 0; margin: 0"> </td>
</tr>
<tr>
 <td width="21%" height="0%" valign="middle" nowrap>
  
 </td>
 <td width="79%" height="0" valign="top">
  
 </td>
</tr>
<tr>
 <td width="21%" height="0%" valign="top" nowrap>
  <table border="0" width="100%">
    <tr>
 <td width="21%" height="0%" valign="top">
  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
      <input type="hidden" name="action" value="home_page">
      <input type="IMAGE" src= "/src/webadmin/myHomePage.gif">

  </form>
 </td>
    </tr>
    <tr>
 <td width="21%" height="0%" valign="top">
  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
      <input type="hidden" name="action" value="department">
      <input type="IMAGE" src= "/src/webadmin/department.gif">
  </form>
 </td>
    </tr>
    <tr>
 <td width="21%" height="0%" valign="top">
  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

            <input type="hidden" name="action" value="personal_information">
      <input type="IMAGE" src= "/src/webadmin/personal.gif">
  </form>
 </td>
    </tr>
    <tr>
 <td width="21%" height="0%" valign="top">
  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
                <input type="hidden" name="action" value="department">
      <input type="IMAGE" src= "/src/webadmin/students.gif">
  </form>
 </td>
    </tr>
    <tr>
    <td width="21%" height="0%" valign="top">
  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

            <input type="hidden" name="action" value="department">
      <input type="IMAGE" src= "/src/webadmin/instructors.gif">
  </form>
 </td>
    </tr>
```

```
      <tr>
      <td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

      <p>
<a     href="mailto:ProjectManager@dho.edu.tr"><img     border="0"     src="/src/webadmin/feedback.gif"
width="190" height="30"></a></p>

     </form>
    </td>
      </tr>
            <tr>
      <td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
            <input type="hidden" name="action" value="log_off">
       <input type="IMAGE" src= "/src/webadmin/log_off.gif">
     </form>
    </td>
      </tr>



 <!--      </tr> -->
     </table>
    </td>
    <td width="79%" height="0" valign="top">
 <!-- this is the special part that the jsp will fill
       according to the operation value
       if operation is announcements excecute the announcements-->

 <% if (myDepHead.getOperation().equals("announcements")) {%>

 <!-- this is the announcement part-->

     <table border="1" width="94%">
      <tr>
       <td width="100%" bgcolor='#FF0000' colspan="3">

     <p align="center"><font color='#FFFF00'><b>Announcements</b></font></p>
       </td>
      </tr>
      <tr>
       <td width="15%" bgcolor="#C0C0C0">Number</td>
       <td width="51%" bgcolor="#C0C0C0">Announcement</td>
       <td width="34%" bgcolor="#C0C0C0">Published Date</td>
      </tr>
      <jsp:getProperty name="myDepHead" property="announcements" />
     </table>
<!-- end of announcement part-->

  <%} if (myDepHead.getOperation().equals("department")) {%>

<!-- this is the department-->

   <table border="1" width="100%" height="112">
  <tr>
```

```
   <td width="100%" colspan="5" bgcolor="#FF0000" height="33">
     <p  align="center"><font  color="#FFFF00"><b>Department  Info  for  <jsp:getProperty
name="myDepHead" property="name" />  <jsp:getProperty name="myDepHead" property="lastName" />
     </b></font></td>
  </tr>
  <tr>
   <td width="20%" height="19"><b>Department Name</b></td>
   <td width="80%" height="19" colspan="4">
     <jsp:getProperty name="myDepHead" property="department" /></td>
  </tr>
  <tr>
   <td  width="100%"  height="17"  colspan="5"  bgcolor="#0000FF"><font
color="#FFFF00"><b>Instructors</b></font></td>
  </tr>
  <tr>
   <td width="20%" height="19"><b>No</b></td>
   <td width="20%" height="19"><b>Instructor Degree</b></td>
   <td width="20%" height="19"><b>Name</b></td>
   <td width="20%" height="19"><b>Last Name</b></td>
   <td width="20%" height="19"><b>Rank</b></td>
  </tr>
  <jsp:getProperty name="myDepHead" property="announcements" />
  <tr>
   <td width="100%" height="19" colspan="5" bgcolor="#0000FF"><font color="#FFFF00"><b>Courses
     offered in this semester</b></font></td>
  </tr>
  <tr>
   <td width="20%" height="19"><b>No</b></td>
   <td width="20%" height="19"><b>Course Id</b></td>
   <td width="20%" height="19"><b>Name</b></td>
   <td width="20%" height="19"><b>Sections taking</b></td>
   <td width="20%" height="19"><b>Instructor giving</b></td>
  </tr>
  <jsp:getProperty name="myDepHead" property="courses" />
 </table>

<!-- end of department head part-->

   <%}  if (myDepHead.getOperation().equals("prof_schedule")) {%>

<!--this is the class schedule part-->

 <table border="1" width="100%" cellspacing="1" height="390" bgcolor="#00FFFF">
       <tr>
         <td width="100%" colspan="7" height="19" bgcolor="#FF0000">
           <font color="#FFFF00">
           <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><b>CLASS
           SCHEDULE</b></td> </font>
       </tr>
       <tr>
         <td width="16%" colspan="2" height="19" bgcolor="#0000FF">
           <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </td>
         <td width="84%" colspan="5" height="19" bgcolor="#0000FF">
           <p  align="center"  style="word-spacing:  0;  margin-top:  0;  margin-bottom:  0"><font  size="1"
color="#FFFF00">DAYS</font></td>
       </tr>
```

```
      <tr>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">No</font></td>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Time</font></td>
        <td width="16%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Monday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Tuesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Wednesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Thursday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Friday</font></td>
      </tr>
      <jsp:getProperty name="myDepHead" property="schedule" />
    </table>

<!--end of the class schedule part -->

  <%} if (myDepHead.getOperation().equals("Personal_information")) {%>

<!-- this is the prof personal information part -->

    <table border="1" width="100%">
      <tr>
        <td width="491" height="41">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">Name         

          <input readonly type="text" name="name" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myDepHead" property="name" />">
          </span> 
          M. 
          <input readonly type="text" name="midname" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myDepHead" property="midName" />">
          </span> 
          Last Name 
          <input readonly type="text" name="lastname" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myDepHead" property="lastName" />">
          </span></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">Degree       
          <input readonly type="text" name="degree" size="20" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myDepHead" property="degree" />">
          </span></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">
```

```
        Department
        <input readonly type="text" name="program" size="32" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myDepHead" property="department" />">
        </span></font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">
        Rank           

        <input readonly type="text" name="rank" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myDepHead" property="rank" />">
        </span> </font>
        </td>
        <td width="98" height="41" bgcolor="#FFFFFF">
        <p align="center">
        <img border="0" src="/src/webadmin/<jsp:getProperty name="myDepHead"
property="login"/>.jpg" width="70" height="70">
        </td>
      </tr>
      <tr>
            <td width="595" colspan="2"> 
        <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
          <input type="text" name="email" size="20" tabindex="1"
          value="<jsp:getProperty name="myDepHead" property="email" />">
          </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
          No <input type="text" name="room" size="20" tabindex="2"
          value="<jsp:getProperty name="myDepHead" property="room" />">
           Tel No <input type="text" name="tel" size="20" tabindex="3"
           value="<jsp:getProperty name="myDepHead" property="tel" />"
           ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
          <input type="text" name="address" size="70" tabindex="4"
          value="<jsp:getProperty name="myDepHead" property="address" />"
          > </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
          <input type="text" name="city" size="20" tabindex="5"
          value="<jsp:getProperty name="myDepHead" property="city" />"
          > Province <input type="text" name="province" size="20" tabindex="6"
          value="<jsp:getProperty name="myDepHead" property="province" />"
          ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2"><input type="submit" value="Update" name="updateButton"></font></p>
                <input type="hidden" name="action" value="personal_update">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
        </form>
      </td>
    </tr>
      <tr>
            <td width="595" colspan="2"> 
            <font color="#FF0000" size="2">Message :
            <jsp:getProperty name="myDepHead" property="message" /></font>

      </td>
```

```
      </tr>

            <tr>
                  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

               <td width="595" colspan="2"> 
          <font size="2"><b>PASSWORD
           CHANGE</b></font>
           <p style="line-height: 100%; margin: 0"> </p>
           <p style="line-height: 100%; margin: 0"><font size="2">Your Login
           Id is
           :            
                    
           <input readonly type="text" name="midname" size="20" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myDepHead" property="login"/>"> </font></p>
           <p style="line-height: 100%; margin: 0"><font size="2">Enter the
           new Password:        <input type="password"
name="firstText" size="20"></font></p>
           <p style="line-height: 100%; margin: 0"><font size="2">Confirm
           your new Password: <input type="password" name="secondText" size="20"></font></p>
                  <p style="line-height: 100%; margin: 0"> </p>
           <p style="line-height: 100%; margin: 0"><font size="2"><input type="submit" value="Change
Password" name="B1"></font></p>
                <input type="hidden" name="action" value="change_Password">
        </td>
        </form>
       </tr>
     </table>
<!-- end of personal information part-->


<!-- end of personal information part-->

     <%} if (myDepHead.getOperation().equals("StudentsForDepHead")) {%>


   <jsp:getProperty name="myDepHead" property="announcements" />



         <table border="1" width="689" height="99">
        <tr>
   <td width="89" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">

     Student No
 </td>
   <td width="107" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">
     Name
 </td>
   <td width="97" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">
     Last Name
 </td>
   <td width="424" height="13" colspan="4" bgcolor="#00FF00">
   <p align="center">----------Grades----------</p>
   </td>
        </tr>
```

```
        <tr>
    <td width="127" height="1" bgcolor="#00FF00" valign="middle">
       First Exam
          </td>
    <td width="127" height="1" bgcolor="#00FF00" valign="middle">
     Second Exam
          </td>
    <td width="55" height="1" bgcolor="#00FF00" valign="middle">
      <p style="word-spacing: 0; margin: 0">Final
          </td>
    <td width="102" height="1" bgcolor="#00FF00" valign="middle">
      <p style="word-spacing: 0; margin: 0">Course Grade
          </td>
        </tr>
        <jsp:getProperty name="myDepHead" property="courses" />
       </table>


<!-- end of students for prof -->
    <%} %>
      <p> 
      <p> 
      <p> 
      <p> 
      <p> 
      <p> </td>
  </tr>
</table>



</body>

</html>
```

7. Source Code of the Faculty JSP File

```
<!--File name: Faculty.jsp
   Purposes : To handle the Faculty users requests.-->

 <%@ page errorPage="ErrorPage.jsp" %>
<HTML>
<HEAD>
<jsp:useBean id="myFaculty" scope="session" class="webadmin.Communicator" />
<jsp:setProperty name="myFaculty" property="*" />
<TITLE>
Faculty
</TITLE>
<base target="_self">
<meta name="Microsoft Border" content="none">
</head>

<body bgcolor="#C8C8C8">
<SCRIPT language=JavaScript>
```

```
var width,height
var image,ext,studentId
var cond1,cond2
function transferview(studentId,mail,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Student Picture</TITLE>"
      var s11="<CENTER>";
    var s15=" </CENTER><p><CENTER><a href='mailto:"+ mail+ "'><font size='2'>" +
        "send mail to " + studentId + "</font></a><p> </p>"

    var s2 ="<IMG SRC='/src/webadmin/" + studentId + ".jpg' BORDER=0>"
    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s2+s15+s3+s4)
    ImageWindow.document.close()
}
</SCRIPT>


<table border="0" width="100%" cellspacing="1" height="0">
 <tr>
  <td width="100%" colspan="2" valign="bottom">
  <p       align="center"><img       border="0"       src="/src/webadmin/academy.gif"       width="770"
height="100"></p></td>

 </tr>
 <tr>
  <td width="100%" colspan="2" valign="bottom"> 
    <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
     <jsp:getProperty name="myFaculty" property="name" />
       <jsp:getProperty name="myFaculty" property="lastName" />
      </p>
    <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
      <jsp:getProperty name="myFaculty" property="time" />
     </p>
    <p style="text-indent: 0; word-spacing: 0; margin: 0"> </td>
 </tr>
 <tr>
  <td width="21%" height="0%" valign="middle" nowrap>
   
  </td>
  <td width="79%" height="0" valign="top">
   
  </td>
 </tr>
 <tr>
  <td width="21%" height="0%" valign="top" nowrap>
```

```
 <table border="0" width="100%">
   <tr>
<td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
     <input type="hidden" name="action" value="home_page">
     <input type="IMAGE" src= "/src/webadmin/myHomePage.gif">
 </form>
</td>
   </tr>
   <tr>
<td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
     <input type="hidden" name="action" value="courses">
     <input type="IMAGE" src= "/src/webadmin/courses.gif">
 </form>
</td>
   </tr>
   <tr>
<td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
          <input type="hidden" name="action" value="personal_information">
     <input type="IMAGE" src= "/src/webadmin/personal.gif">
 </form>
</td>
   </tr>
   <tr>
<td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
                <input type="hidden" name="action" value="schedule">
     <input type="IMAGE" src= "/src/webadmin/classSchedule.gif">
 </form>
</td>
   </tr>
   <tr>
   <td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
          <input type="hidden" name="action" value="courses">
     <input type="IMAGE" src= "/src/webadmin/students.gif">
 </form>
</td>
   </tr>
   <tr>
<td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
     <input type="hidden" name="action" value="courses">
     <input type="IMAGE" src= "/src/webadmin/submitGrades.gif">
 </form>
</td>
   </tr>
   <tr>
   <tr>
   <td width="21%" height="0%" valign="top">
 <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

  <p>
```

```
<a href="mailto:ProjectManager@dho.edu.tr"><img border="0" src="/src/webadmin/feedback.gif"
width="190" height="30"></a></p>

    </form>
   </td>
     </tr>
          <tr>
     <td width="21%" height="0%" valign="top">
    <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
          <input type="hidden" name="action" value="log_off">
      <input type="IMAGE" src= "/src/webadmin/log_off.gif">
    </form>
   </td>
     </tr>
<!--      </tr> -->
    </table>
   </td>
   <td width="79%" height="0" valign="top">
<!-- this is the special part that the jsp will fill
       according to the operation value
       if operation is announcements excecute the announcements-->

<% if (myFaculty.getOperation().equals("announcements")) {%>

<!-- this is the announcement part-->

    <table border="1" width="94%">
<tr>
       <td width="100%" bgcolor='#FF0000' colspan="3">

    <p align="center"><font color='#FFFF00'><b>Announcements</b></font></p>
       </td>
      </tr>
      <tr>
       <td width="15%" bgcolor="#C0C0C0">Number</td>
       <td width="51%" bgcolor="#C0C0C0">Announcement</td>
       <td width="34%" bgcolor="#C0C0C0">Published Date</td>
      </tr>
      <jsp:getProperty name="myFaculty" property="announcements" />
    </table>
<!-- end of announcement part-->

 <%} if (myFaculty.getOperation().equals("courses")) {%>

<!-- this is the courses part-->

   <table border="1" width="99%">
      <tr>
       <td width="100%" bgcolor="#FF0000" colspan="6">
         <p align="center"><font color="#FFFF00" ><b>You are currently giving the
         following Courses</b></font></p>
       </td>
      </tr>
      <tr>
       <td width="8%" bgcolor="#C0C0C0"><font size="2">Selection</font></td>
       <td width="9%" bgcolor="#C0C0C0"><font size="2">Course Id</font></td>
```

```
        <td width="11%" bgcolor="#C0C0C0"><font size="2">Course Name</font></td>
        <td width="40%" bgcolor="#C0C0C0"><font size="2">Explanation </font></td>
        <td width="13%" bgcolor="#C0C0C0"><font size="2">Credits</font></td>
        <td width="9%" bgcolor="#C0C0C0"><font size="2">Section</font></td>
       </tr>
       <jsp:getProperty name="myFaculty" property="courses" />
     </table>




<!-- end of courses part-->

   <%}  if (myFaculty.getOperation().equals("prof_schedule")) {%>

<!--this is the class schedule part-->

 <table border="1" width="100%" cellspacing="1" height="390" bgcolor="#00FFFF">
       <tr>
        <td width="100%" colspan="7" height="19" bgcolor="#FF0000">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0">CLASS
          SCHEDULE</td>
       </tr>
       <tr>
        <td width="16%" colspan="2" height="19" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </td>
        <td width="84%" colspan="5" height="19" bgcolor="#0000FF">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="1"
color="#FFFF00">DAYS</font></td>
       </tr>
       <tr>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">No</font></td>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Time</font></td>
        <td width="16%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Monday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Tuesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Wednesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Thursday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Friday</font></td>
       </tr>
       <jsp:getProperty name="myFaculty" property="schedule" />
     </table>
```

```
<!--end of the class schedule part -->

  <%} if (myFaculty.getOperation().equals("Personal_information")) {%>

<!-- this is the prof personal information part -->

    <table border="1" width="100%">
      <tr>
        <td width="491" height="41">
         <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
         <font size="2">Name         

         <input readonly type="text" name="name" size="12" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="name" />">
         </span> 
         M. 
         <input readonly type="text" name="midname" size="12" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="midName" />">
         </span> 
         Last Name 
         <input readonly type="text" name="lastname" size="12" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="lastName" />">
         </span></font></p>
         <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
         <font size="2">Degree       
         <input readonly type="text" name="degree" size="20" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="degree" />">
         </span></font></p>
         <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
         <font size="2">
         Department
         <input readonly type="text" name="program" size="32" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="department" />">
         </span></font></p>
         <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">
         Rank           

         <input readonly type="text" name="rank" size="12" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="rank" />">
         </span> </font>
         </td>
         <td width="98" height="41" bgcolor="#FFFFFF">
         <p align="center">
         <img border="0" src="/src/webadmin/<jsp:getProperty name="myFaculty" property="login"/>.jpg"
width="70" height="70">
         </td>
      </tr>
      <tr>
              <td width="595" colspan="2"> 
         <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
           <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
           <input type="text" name="email" size="20" tabindex="1"
           value="<jsp:getProperty name="myFaculty" property="email" />">
           </font></p>
```

```
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
          No <input type="text" name="room" size="20" tabindex="2"
          value="<jsp:getProperty name="myFaculty" property="room" />">
           Tel No <input type="text" name="tel" size="20" tabindex="3"
           value="<jsp:getProperty name="myFaculty" property="tel" />"
           ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
          <input type="text" name="address" size="70" tabindex="4"
          value="<jsp:getProperty name="myFaculty" property="address" />"
          > </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
          <input type="text" name="city" size="20" tabindex="5"
          value="<jsp:getProperty name="myFaculty" property="city" />"
          > Province <input type="text" name="province" size="20" tabindex="6"
          value="<jsp:getProperty name="myFaculty" property="province" />"
          ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2"><input type="submit" value="Update" name="updateButton"></font></p>
                <input type="hidden" name="action" value="personal_update">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
         </form>
        </td>
       </tr>
         <tr>
              <td width="595" colspan="2"> 
              <font color="#FF0000" size="2">Message :
              <jsp:getProperty name="myFaculty" property="message" /></font>

        </td>
       </tr>

            <tr>
                  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

              <td width="595" colspan="2"> 
          <font size="2"><b>PASSWORD
          CHANGE</b></font>
          <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2">Your Login
          Id is
          :           
                   
          <input readonly type="text" name="midname" size="20" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myFaculty" property="login"/>"> </font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Enter the
          new Password:        <input type="password"
name="firstText" size="20"></font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Confirm
          your new Password: <input type="password" name="secondText" size="20"></font></p>
                  <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2"><input type="submit" value="Change
Password" name="B1"></font></p>
               <input type="hidden" name="action" value="change_Password">
       </td>
```

```
        </form>
       </tr>
      </table>
<!-- end of personal information part-->
<!-- this is the students for prof part -->
     <%} if (myFaculty.getOperation().equals("StudentsForProfessor")) {%>


   <jsp:getProperty name="myFaculty" property="announcements" />


       <table border="1" width="689" height="99">
        <tr>
   <td width="84" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">

     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
       <p style="word-spacing: 0; margin: 0"><font size="3"><input type="submit" value="Student No"
name="B2" style="font-size: 8pt"></font></p>
       <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="student_no">
       <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
       <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
 </td>
   <td width="140" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">
     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
       <p style="word-spacing: 0; margin: 0"><font size="3"><input type="submit" value="Name"
name="B2" style="font-size: 8pt"></font></p>
       <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="name">
       <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
       <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
 </td>
   <td width="99" height="1" rowspan="2" bgcolor="#00FF00" valign="middle">
     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
       <p style="word-spacing: 0; margin: 0"><font size="3"><input type="submit" value="Last Name"
name="B2" style="font-size: 8pt"></font></p>
       <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="last_name">
       <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
       <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
 </td>
   <td width="424" height="13" colspan="4" bgcolor="#00FF00">
   <p align="center">----------Grades----------</p>
   </td>
```

```
        </tr>
        <tr>
   <td width="111" height="1" bgcolor="#00FF00" valign="middle">
      <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
      <p style="word-spacing: 0; margin: 0"><input type="submit" value="First Exam" name="B1"
style="font-size: 8pt"></p>
       <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="first_midterm">
      <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
      <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
         </td>
   <td width="134" height="1" bgcolor="#00FF00" valign="middle">
     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
      <p style="word-spacing: 0; margin: 0"><input type="submit" value="Second Exam" name="B1"
style="font-size: 8pt"></p>
      <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="second_midterm">
      <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
      <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
         </td>
   <td width="65" height="1" bgcolor="#00FF00" valign="middle">
     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
      <p style="word-spacing: 0; margin: 0"><input type="submit" value="Final" name="B2" style="font-
size: 8pt"></p>
      <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="final">
      <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
      <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
         </td>
   <td width="99" height="1" bgcolor="#00FF00" valign="middle">
     <form method="POST" style="word-spacing: 0; margin: 0" action="/servlet/webadmin.Academy"
onSubmit="">
      <p style="word-spacing: 0; margin: 0"><input type="submit" value="Course Grade" name="B1"
style="font-size: 8pt"></p>
      <input type="hidden" name="action" value="studentsForProf"><input type="hidden" name="order"
value="grade">
      <input type="hidden" name="course" value="<jsp:getProperty name="myFaculty" property="course"
/>">
      <input type="hidden" name="section" value="<jsp:getProperty name="myFaculty"
property="section" />">
     </form>
         </td>
        </tr>
      </table>
```

```
        <jsp:getProperty name="myFaculty" property="courses" />
<!-- end of students for prof -->
   <%} %>




    <p> 
    <p> 
    <p> 
    <p> 
    <p> 
    <p> </td>
  </tr>
</table>



</body>

</html>
```

8. Source Code of the Regiment JSP File

```
<!--File name: Regiment.jsp
   Purposes : To handle the Regiment users requests.-->

 <%@ page errorPage="ErrorPage.jsp" %>
<HTML>
<HEAD>
<jsp:useBean id="myRegiment" scope="session" class="webadmin.Communicator" />
<jsp:setProperty name="myRegiment" property="*" />
<TITLE>
Regiment
</TITLE>
<base target="_self">
<meta name="Microsoft Border" content="none">
</head>

<body bgcolor="#C8C8C8">
<SCRIPT language=JavaScript>

var width,height
var image,ext,studentId
var cond1,cond2
function facultyview(name,lastName,login,mail,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Faculty info</TITLE>"
      var s11="<CENTER>";
    var s2 ="<IMG SRC='/src/webadmin/" + login + ".jpg' BORDER=0>"
```

```
    var s14=" </CENTER><p><CENTER><font size='2'>" + name +
        " " + lastName + "</font></a>"

    var s15=" </CENTER><p><CENTER><a href='mailto:"+ mail+ "'><font size='2'>" +
        "send mail to " + login + "</font></a>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s2+s14+s15+s3+s4)
    ImageWindow.document.close()
}

var width,height
var cId,cName,cCredit,ext,studentId
var cond1,cond2
function courseview(cId,cName,cCredit,cExp,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Course info</TITLE>"
      var s11="<CENTER>";

      var s12 = "<table border='1' width='74%' height='106'> <tr>"
      var s13 = "<td width='41%' height='47'><b>Course id      : </b>"+
cId + "</td>"
      var s14 = "<td width='59%' height='141' rowspan='3'><b>Course Explanation : </b>" +cExp + "</td>
</tr>  <tr>"
      var s15 = "<td width='41%' height='47'><b>Course credit: </b>" + cCredit + "</td>  </tr>  <tr>"
      var s16 = "<td width='41%' height='47'><b>Course name: </b>" + cName + "</td> </tr></table>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s12+s13+s14+s15+s16+s3+s4)
    ImageWindow.document.close()
}
</SCRIPT>

<table border="0" width="100%" cellspacing="1" height="0">
  <tr>
    <td width="100%" colspan="2" valign="bottom">
    <p       align="center"><img       border="0"       src="/src/webadmin/academy.gif"       width="770"
height="100"></p></td>

  </tr>
  <tr>
    <td width="100%" colspan="2" valign="bottom"> 
```

```
    <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
     <jsp:getProperty name="myRegiment" property="name" />
       <jsp:getProperty name="myRegiment" property="lastName" />
      </p>
    <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
      <jsp:getProperty name="myRegiment" property="time" />
     </p>
    <p style="text-indent: 0; word-spacing: 0; margin: 0"> </td>
</tr>
<tr>
  <td width="21%" height="0%" valign="middle" nowrap>
   
  </td>
  <td width="79%" height="0" valign="top">
   
  </td>
</tr>
<tr>
  <td width="21%" height="0%" valign="top" nowrap>
    <table border="0" width="100%">
      <tr>
  <td width="21%" height="0%" valign="top">
   <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
      <p>
        <input type="hidden" name="action" value="home_page">
        <input type="IMAGE" src= "/src/webadmin/myHomePage.gif">
        </p>

    </form>
  </td>
      </tr>
      <tr>
  <td width="21%" height="0%" valign="top">
   <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
        <input type="hidden" name="action" value="unitsForRegiment">
        <input type="IMAGE" src= "/src/webadmin/units.gif">

    </form>
  </td>
      </tr>
      <tr>
  <td width="21%" height="0%" valign="top">
    <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
      <p><font size="1">       </font></p>
              <input type="hidden" name="action" value="personal_information">
              <input type="IMAGE" src= "/src/webadmin/personal.gif">
    </form>
  </td>
      </tr>
      <tr>
  <td width="21%" height="0%" valign="top">
    <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
      <p><font size="1"><input type="IMAGE" src= "/src/webadmin/studentsOfficers.gif"></font></p>
                  <input type="hidden" name="action" value="unitsForRegiment">
                 <!--here the user asks for the class schedule-->
    </form>
```

```
   </td>
      </tr>
      <tr>
      <td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

      <p>
<a     href="mailto:ProjectManager@dho.edu.tr"><img     border="0"     src="/src/webadmin/feedback.gif"
width="190" height="30"></a></p>

     </form>
   </td>
      </tr>
                  <tr>
      <td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
             <input type="hidden" name="action" value="log_off">
       <input type="IMAGE" src= "/src/webadmin/log_off.gif">
     </form>
   </td>
      </tr>

 <!--      </tr> -->
     </table>
   </td>
   <td width="79%" height="0" valign="top">
 <!-- this is the special part that the jsp will fill
       according to the operation value
       if operation is announcements excecute the announcements-->

 <% if (myRegiment.getOperation().equals("announcements")) {%>

 <!-- this is the announcement part-->

     <table border="3" width="94%" bordercolor="#0000FF">
       <tr>
         <td width="100%" bgcolor='#FF0000' colspan="3">

     <p align="center"><font color='#FFFF00'><b>Announcements</b></font></p>
        </td>
       </tr>
       <tr>
         <td width="15%" bgcolor="#C0C0C0"><b>Number</b></td>
         <td width="51%" bgcolor="#C0C0C0"><b>Announcement</b></td>
         <td width="34%" bgcolor="#C0C0C0"><b>Published Date</b></td>
       </tr>
       <jsp:getProperty name="myRegiment" property="announcements" />
     </table>
<!-- end of announcement part-->

  <%} if (myRegiment.getOperation().equals("sectionInfoForRegiment")) {%>

<!-- this is the courses part-->


<table border='3' width='100%' height='90' bordercolor='#0000FF'>
```

```
 <tr>
  <td width='100%' height='40' bgcolor='#FF0000' colspan='5'>
   <p align='center'><font color='#FFFF00'><b>The Section Information for
   <jsp:getProperty name="myRegiment" property="regComName" /> </b></font></td>
 </tr>
 <tr>
  <td width='39%' height='19' bgcolor='#0000FF' colspan='3'><font color='#FFFFFF'><b>Company
name:</b></font> </td>
  <td width='61%' height='19' bgcolor='#0000FF' colspan='2'><font color='#FFFFFF'>
  <jsp:getProperty name="myRegiment" property="companyName" /></font></td>
 </tr>
 <tr>
  <td width='39%' height='19' bgcolor='#0000FF' colspan='3'><font color='#FFFFFF'><b>Section
name:</b></font> </td>
  <td width='61%' height='19' bgcolor='#0000FF' colspan='2'><font color='#FFFFFF'>
  <jsp:getProperty name="myRegiment" property="section" /></font></td>
 </tr>
 <tr>
  <td width='39%' height='1' colspan='3' bgcolor='#0000FF'><font color='#FFFFFF'><b>Number
   of students in the section:</b></font></td>
  <td width='61%' height='1' colspan='2' bgcolor='#0000FF'><font color='#FFFFFF'>
  <jsp:getProperty name="myRegiment" property="awards" /></font></td>
 </tr>
  <tr>
  <td width='20%' height='1' bgcolor='#C0C0C0'><b>Student No</b></td>
  <td width='20%' height='1' bgcolor='#C0C0C0'><b>Name</b></td>
  <td width='20%' height='1' colspan='2' bgcolor='#C0C0C0'><b>Last Name</b></td>
  <td width='20%' height='1' bgcolor='#C0C0C0'><b>Discipline points</b></td><b>
 </tr>


     <jsp:getProperty name="myRegiment" property="regiment" />
    </table>




<!-- end of courses part-->

  <%}  if (myRegiment.getOperation().equals("unitsForRegiment")) {%>

<!--this is the regiment unit info part-->
<table border="3" width="100%" height="153" bordercolor="#0000FF">
 <tr>
  <tr>
  <td width="100%" height="40" bgcolor="#FF0000" colspan="2">
   <p align="center"><font color="#FFFF00"><b>The Units Commanded by <jsp:getProperty
name="myRegiment" property="regComName" /> 
   </b></font>
   <p align="center"><font color="#FFFF00"><b>"Commander of the <jsp:getProperty
name="myRegiment" property="companyName" />"</b></font></td>
 </tr>


   <jsp:getProperty name="myRegiment" property="regiment" />
```

```
    <table>

<!--end of the regiment unit info part -->

  <%} if (myRegiment.getOperation().equals("Personal_information")) {%>

     <table border="3" width="100%" bordercolor="#0000FF">
       <tr>
         <td width="491" height="41">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">Name         

          <input readonly type="text" name="name" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myRegiment" property="name1" />">
          </span> 
          M. 
          <input readonly type="text" name="midname" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myRegiment" property="midName1" />">
          </span> 
          Last Name 
          <input readonly type="text" name="lastname" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myRegiment" property="lastName1" />">
          </span></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">
          Unit Commanded
          <input readonly type="text" name="program" size="40" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myRegiment" property="companyName" />">
          </span></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">
          Rank           

          <input readonly type="text" name="rank" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myRegiment" property="rank" />">
          </span> </font>
          </td>
          <td width="98" height="41" bgcolor="#FFFFFF">
          <p align="center">
          <img border="0" src="/src/webadmin/<jsp:getProperty name="myRegiment"
property="login1"/>.jpg" width="70" height="70">
          </td>
       </tr>
       <tr>
              <td width="595" colspan="2"> 
          <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
            <input type="text" name="email" size="20" tabindex="1"
            value="<jsp:getProperty name="myRegiment" property="email" />">
            </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
            No <input type="text" name="room" size="20" tabindex="2"
            value="<jsp:getProperty name="myRegiment" property="room" />">
             Tel No <input type="text" name="tel" size="20" tabindex="3"
             value="<jsp:getProperty name="myRegiment" property="tel" />"
             ></font></p>
```

```
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
          <input type="text" name="address" size="70" tabindex="4"
          value="<jsp:getProperty name="myRegiment" property="address" />"
          > </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
          <input type="text" name="city" size="20" tabindex="5"
          value="<jsp:getProperty name="myRegiment" property="city" />"
          > Province <input type="text" name="province" size="20" tabindex="6"
          value="<jsp:getProperty name="myRegiment" property="province" />"
          ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2"><input type="submit" value="Update" name="updateButton"></font></p>
                <input type="hidden" name="action" value="personal_update">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
         </form>
        </td>
       </tr>
        <tr>
              <td width="595" colspan="2"> 
              <font color="#FF0000" size="2">Message :
              <jsp:getProperty name="myRegiment" property="message" /></font>

        </td>
       </tr>

            <tr>
                  <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

              <td width="595" colspan="2"> 
         <font size="2"><b>PASSWORD
          CHANGE</b></font>
          <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2">Your Login
          Id is
          :           
                   
          <input readonly type="text" name="midname" size="20" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myRegiment" property="login"/>"> </font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Enter the
          new Password:        <input type="password"
name="firstText" size="20"></font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Confirm
          your new Password: <input type="password" name="secondText" size="20"></font></p>
                  <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2"><input type="submit" value="Change
Password" name="B1"></font></p>
              <input type="hidden" name="action" value="change_Password">
       </td>
       </form>
      </tr>
    </table>
<!-- end of personal information part-->

  <%} if (myRegiment.getOperation().equals("commanderInfoForRegiment")) {%>
```

```
    <table border="3" width="100%" bordercolor="#0000FF">
      <tr>
        <td width="589" height="41" colspan="2" bgcolor="#FF0000">
        <p align='center'>
         <font color="#FFFF00"><b>Extra information about <jsp:getProperty name="myRegiment"
property="rank" />
          <jsp:getProperty name="myRegiment" property="name1" /> <jsp:getProperty
name="myRegiment" property="lastName1" /></b></font></p>
        </td>
      </tr>

      <tr>
        <td width="491" height="41">
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
        <font size="2">Name         

        <input readonly type="text" name="name" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="name1" />">
        </span> 
        M. 
        <input readonly type="text" name="midname" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="midName1" />">
        </span> 
        Last Name 
        <input readonly type="text" name="lastname" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="lastName1" />">
        </span></font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
        <font size="2">
        Unit Commanded
        <input readonly type="text" name="program" size="40" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="companyName" />">
        </span></font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">
        Rank           

        <input readonly type="text" name="rank" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="rank" />">
        </span> </font>
        </td>
        <td width="98" height="41" bgcolor="#FFFFFF">
        <p align="center">
        <img border="0" src="/src/webadmin/<jsp:getProperty name="myRegiment"
property="login1"/>.jpg" width="70" height="70">
        </td>
      </tr>
      <tr>
            <td width="595" colspan="2"> 

          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
          <input type="text" name="email" size="20" style="background-color: #C0C0C0"tabindex="1"
          value="<jsp:getProperty name="myRegiment" property="email" />">
          <a href="mailto:<jsp:getProperty name="myRegiment" property="email" />">sendMail</a>
          </font></p>
```

```
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
          No <input type="text" name="room" size="20" style="background-color: #C0C0C0"
tabindex="2"
          value="<jsp:getProperty name="myRegiment" property="room" />">
           Tel No <input type="text" name="tel" size="20" style="background-color: #C0C0C0"
tabindex="3"
           value="<jsp:getProperty name="myRegiment" property="tel" />"
           ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
          <input type="text" name="address" size="70" style="background-color: #C0C0C0"tabindex="4"
          value="<jsp:getProperty name="myRegiment" property="address" />"
          > </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
          <input type="text" name="city" size="20" style="background-color: #C0C0C0"tabindex="5"
          value="<jsp:getProperty name="myRegiment" property="city" />"
          > Province <input type="text" name="province" size="20" style="background-color:
#C0C0C0"tabindex="6"
          value="<jsp:getProperty name="myRegiment" property="province" />"
          ></font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>


        </td>

    </table>
<!-- end of commander info part-->



<!-- this is the showcompany info part -->
    <%} if (myRegiment.getOperation().equals("showCompanyInfo")) {%>



<table border='3' width='100%' height='90' bordercolor='#0000FF'>
 <tr>
  <td width='100%' height='40' bgcolor='#FF0000' colspan='2'>
   <p align='center'><font color='#FFFF00'><b>The Company Information for
  <jsp:getProperty name="myRegiment" property="regComName" /> </b></font></td>
 </tr>
 <tr>
  <td width='39%' height='19' bgcolor='#0000FF'><b><font color='#FFFFFF'>Company
   name:</font> </b></td>
  <td width='61%' height='19' bgcolor='#0000FF'><font color='#FFFFFF'>
  <jsp:getProperty name="myRegiment" property="companyName" /></font></td>
 </tr>
 <tr>
 <form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=''>
  <td width='39%' height='1'><b>Company Commander: </b></td>
  <td width='61%' height='1'>
  <input type='submit' value='<jsp:getProperty name="myRegiment" property="compComName" />'
name='B1'></td>
  <input type='hidden' name='action' value='showCommanderInfo'>
```

```
   <input type='hidden' name='person_id' value='<jsp:getProperty name="myRegiment" property="userId"
/>'>
   <input type='hidden' name='person_loginid' value='<jsp:getProperty name="myRegiment"
property="login1" />'>
  </form>
 </tr>
 <tr>
 <form method='POST' style='word-spacing: 0; margin: 0' action='/servlet/webadmin.Academy'
onSubmit=''>
   <td width='39%' height='1'><b>Sections of the Company:</b></td>
   <td width='61%' height='1'><input type='submit' value='                    <jsp:getProperty
name="myRegiment" property="section" />              ' name='section'>
    <p><input type='submit' value='              <jsp:getProperty name="myRegiment" property="course" />
' name='section'></td>
       <input type='hidden' name='action' value='showSectionInfo'>
 </form>
 </tr>
</table>

<!-- end of companyinfo  -->
 <%} if (myRegiment.getOperation().equals("showStudentInformation")) {%>

<!-- this is the refgiment personal information part -->

    <table border="3" width="100%" height="542" bordercolor="#0000FF">
     <tr>
       <td width="589" height="36" colspan="2" bgcolor="#FF0000">
        <p align="center"><font color="#FFFF00"><b>The Information about Student  
        <jsp:getProperty name="myRegiment" property="name1" /> 
        <jsp:getProperty name="myRegiment" property="lastName1" />
        </b></font>
        </td>
     </tr>
     <tr>
       <td width="491" height="45">
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
         </p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
        <font size="2">Name          
        <input type="text" name="name" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="name1" />">  
        M. 
        <input type="text" name="midname" size="12" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="midName1" />">
         
        Last Name  <input type="text" name="lastname" size="12" style="background-color:
#C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="lastName1" />">
        </font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Section</font><font
size="2">
               <input type="text" name="degree" size="20"
style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="section" />">
        </font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
```

```
        <font size="2">
        Department
        <input type="text" name="program" size="51" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="department" />">
        </font></p>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Regiment
        Unit <input type="text" name="rank" size="50" style="background-color: #C0C0C0"
        value="<jsp:getProperty name="myRegiment" property="regiment" />">
         </font>
        <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> 
        </td>
        <td width="98" height="92" bgcolor="#FFFFFF">
        <p align="center">
        <img border="0" src="/src/webadmin/<jsp:getProperty name="myRegiment"
property="userId"/>.jpg" width="70" height="70">
        </td>
      </tr>
      <tr>
            <td width="595" colspan="2" height="1"> 
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
          <input type="text" name="email" size="20" style="background-color: #C0C0C0"  tabindex="1"
          value="<jsp:getProperty name="myRegiment" property="email" />">
          <a href="mailto:<jsp:getProperty name="myRegiment" property="email" />">sendMail</a>
          </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Dormitory:  </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
          No <input type="text" name="room" size="20" style="background-color: #C0C0C0"
tabindex="2"
          value="<jsp:getProperty name="myRegiment" property="room" />">
          Location <input type="text" name="LOCATION" size="20" style="background-color: #C0C0C0"
tabindex="3"
           value="<jsp:getProperty name="myRegiment" property="location" />">
           </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Home:</font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
          <input type="text" name="address" size="70" style="background-color: #C0C0C0" tabindex="4"
          value="<jsp:getProperty name="myRegiment" property="address" />">
           </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
          <input type="text" name="city" size="20" style="background-color: #C0C0C0" tabindex="5"
          value="<jsp:getProperty name="myRegiment" property="city" />">
           Province <input type="text" name="province" size="20" style="background-color: #C0C0C0"
tabindex="6"
          value="<jsp:getProperty name="myRegiment" property="province" />">
          </font></p>


        </td>
      </tr>
      </table>
```

```
<table border="3" width="100%" bordercolor="#0000FF">
<tr>
   <td width="679" colspan="5" bgcolor="#0000FF" height="40">
     <p align="center"><font color="#FFFF00"><b>AWARDS</b></font></td>
 </tr>
 <tr>
   <td width="84" height="19"><b>Award No</b></td>
   <td width="98" height="19"><b>Award Type</b></td>
   <td width="248" height="19"><b>Result</b></td>
   <td width="231" height="19" colspan="2"><b>Award Date</b></td>
 </tr>
 <jsp:getProperty name="myRegiment" property="awards" />
   <tr>
   <td width="679" colspan="5" bgcolor="#0000FF" height="40">
     <p align="center"><font color="#FFFF00"><b>PUNISHMENTS</b></font></td>
 </tr>
 <tr>
   <td width="84" height="19"><b>Punishment No</b></td>
   <td width="98" height="19"><b>Punishment Type</b></td>
   <td width="248" height="19"><b>Result</b></td>
   <td width="126" height="19"><b>Points Taken</b></td>
   <td width="99" height="19"><b>Punishment Date</b></td>
 </tr>
 <jsp:getProperty name="myRegiment" property="punishments" />
 </table>
<table border="3" width="689" height="77">
        <tr>
   <td width="92" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">

     <font color="#FFFF00"><b>Professor</b></font>
 </td>
   <td width="124" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">
     <font color="#FFFF00"><b>Course Id</b></font>
 </td>
   <td width="107" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">
     <font color="#FFFF00"><b>Course Name</b></font>
 </td>
   <td width="424" height="13" colspan="4" bgcolor="#0000FF">
   <p          align="center"><font          color="#FFFF00"><b>----------------------Grades-------------------------
</b></font></p>
   </td>
        </tr>
        <tr>
   <td width="111" height="1" bgcolor="#0000FF" valign="middle">
      <font color="#FFFF00"><b>First Exam</b></font>
         </td>
   <td width="134" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Second Exam</b></font>
         </td>
   <td width="65" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Final</b></font>
         </td>
   <td width="99" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Course Grade</b></font>
         </td>
        </tr>
```

```
   <jsp:getProperty name="myRegiment" property="courses" />
    </table>
 <table border="3" width="100%" cellspacing="1" height="390" bgcolor="#00FFFF">
       <tr>
        <td width="100%" colspan="7" height="19" bgcolor="#FF0000">
          <font color="#FFFF00">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><b>CLASS
          SCHEDULE</b></td> </font>
       </tr>
       <tr>
        <td width="16%" colspan="2" height="19" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </td>
        <td width="84%" colspan="5" height="19" bgcolor="#0000FF">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="1"
color="#FFFF00">DAYS</font></td>
       </tr>
       <tr>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">No</font></td>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Time</font></td>
        <td width="16%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Monday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Tuesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Wednesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Thursday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Friday</font></td>
       </tr>
       <jsp:getProperty name="myRegiment" property="schedule" />
     </table>


     <%} %>


</body>

</html>
```

9. Source Code of the Student JSP File

```
<!--File name: Student.jsp
   Purposes : To handle the Student users requests.-->

 <%@ page errorPage="ErrorPage.jsp" %>
<HTML>
<HEAD>
<jsp:useBean id="myStudent" scope="session" class="webadmin.Communicator" />
<jsp:setProperty name="myStudent" property="*" />
<TITLE>
Student
</TITLE>
<base target="_self">
<meta name="Microsoft Border" content="none">
</head>

<body bgcolor="#C8C8C8">
<SCRIPT language=JavaScript>

var width,height
var image,ext,studentId
var cond1,cond2
function facultyview(name,lastName,login,mail,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Faculty info</TITLE>"
      var s11="<CENTER>";
    var s2 ="<IMG SRC='/src/webadmin/" + login + ".jpg' BORDER=0>"
    var s14=" </CENTER><p><CENTER><font size='2'>" + name +
        " " + lastName + "</font></a>"

    var s15=" </CENTER><p><CENTER><a href='mailto:"+ mail+ "'><font size='2'>" +
        "send mail to " + login + "</font></a>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s2+s14+s15+s3+s4)
    ImageWindow.document.close()
}

var width,height
var cId,cName,cCredit,ext,studentId
var cond1,cond2
function courseview(cId,cName,cCredit,cExp,width,height) {
    if (width==0) cond1=" "
        else cond1="width="+(width+20)+"";
    if (height==0) {cond2=" "};
```

```
        else {cond2="height="+(height+70)+""};

    var s1 ="<TITLE>Course info</TITLE>"
      var s11="<CENTER>";

      var s12 = "<table border='1' width='74%' height='106'> <tr>"
      var s13 = "<td width='41%' height='47'><b>Course id      : </b>"+
cId + "</td>"
      var s14 = "<td width='59%' height='141' rowspan='3'><b>Course Explanation : </b>" +cExp + "</td>
</tr>  <tr>"
      var s15 = "<td width='41%' height='47'><b>Course credit: </b>" + cCredit + "</td>  </tr>  <tr>"
      var s16 = "<td width='41%' height='47'><b>Course name: </b>" + cName + "</td> </tr></table>"

    var s3 ="<FORM><INPUT TYPE='BUTTON' VALUE='Close Window'"+     "
onClick='self.close()'>"
    var s4 ="</FORM></CENTER>"

    ImageWindow=window.open("",
"newwin"+width,"toolbar=no,scrollbars="+scroll+",menubar=no,"+cond1+","+cond2);
    ImageWindow.document.write(s1+s11+s12+s13+s14+s15+s16+s3+s4)
    ImageWindow.document.close()
}
</SCRIPT>

<table border="0" width="100%" cellspacing="1" height="0">
 <tr>
  <td width="100%" colspan="2" valign="bottom">
  <p        align="center"><img        border="0"        src="/src/webadmin/academy.gif"        width="770"
height="100"></p></td>

 </tr>
 <tr>
  <td width="100%" colspan="2" valign="bottom"> 
   <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
    <jsp:getProperty name="myStudent" property="name" />
      <jsp:getProperty name="myStudent" property="lastName" />
     </p>
   <p style="text-indent: 0; word-spacing: 0; margin: 0" align="center">
     <jsp:getProperty name="myStudent" property="time" />
    </p>
   <p style="text-indent: 0; word-spacing: 0; margin: 0"> </td>
 </tr>
 <tr>
  <td width="21%" height="0%" valign="middle" nowrap>
   
  </td>
  <td width="79%" height="0" valign="top">
   
  </td>
 </tr>
 <tr>
  <td width="21%" height="0%" valign="top" nowrap>
   <table border="0" width="100%">
     <tr>
  <td width="21%" height="0%" valign="top">
   <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
```

```
        <input type="hidden" name="action" value="home_page">
        <input type="IMAGE" src= "/src/webadmin/myHomePage.gif">

      </form>
    </td>
      </tr>
      <tr>
    <td width="21%" height="0%" valign="top">
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
        <input type="hidden" name="action" value="courses">
        <input type="IMAGE" src= "/src/webadmin/courses.gif">
      </form>
    </td>
      </tr>
      <tr>
    <td width="21%" height="0%" valign="top">
      <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
              <input type="hidden" name="action" value="personal_information">
        <input type="IMAGE" src= "/src/webadmin/personal.gif">
      </form>
    </td>
      </tr>
      <tr>
    <td width="21%" height="0%" valign="top">
      <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
                    <input type="hidden" name="action" value="schedule">
        <input type="IMAGE" src= "/src/webadmin/classSchedule.gif">
      </form>
    </td>
      </tr>
      <tr>
      <td width="21%" height="0%" valign="top">
      <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
              <input type="hidden" name="action" value="gradesForStudent">
        <input type="IMAGE" src= "/src/webadmin/grades.gif">
      </form>
    </td>
      </tr>
        <tr>
      <td width="21%" height="0%" valign="top">
      <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
              <input type="hidden" name="action" value="companyInfoForStudent">
        <input type="IMAGE" src= "/src/webadmin/company.gif">
      </form>
    </td>
      </tr>
      <tr>
      <td width="21%" height="0%" valign="top">
      <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

       <p>
<a     href="mailto:ProjectManager@dho.edu.tr"><img     border="0"     src="/src/webadmin/feedback.gif"
width="190" height="30"></a></p>

      </form>
    </td>
```

```
      </tr>
            <tr>
      <td width="21%" height="0%" valign="top">
    <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
            <input type="hidden" name="action" value="log_off">
       <input type="IMAGE" src= "/src/webadmin/log_off.gif">
    </form>
   </td>
      </tr>



<!--      </tr> -->
    </table>
   </td>
   <td width="79%" height="0" valign="top">
<!-- this is the special part that the jsp will fill
       according to the operation value
       if operation is announcements excecute the announcements-->

<% if (myStudent.getOperation().equals("announcements")) {%>

<!-- this is the announcement part-->

    <table border="1" width="94%">
      <tr>
       <td width="100%" bgcolor='#FF0000' colspan="3">

    <p align="center"><font color='#FFFF00'><b>Announcements</b></font></p>
       </td>
      </tr>
      <tr>
       <td width="15%" bgcolor="#C0C0C0">Number</td>
       <td width="51%" bgcolor="#C0C0C0">Announcement</td>
       <td width="34%" bgcolor="#C0C0C0">Published Date</td>
      </tr>
      <jsp:getProperty name="myStudent" property="announcements" />
    </table>
<!-- end of announcement part-->

 <%} if (myStudent.getOperation().equals("courses")) {%>

<!-- this is the courses part-->

   <table border="1" width="99%">
      <tr>
       <td width="100%" bgcolor="#FF0000" colspan="6">
         <p align="center"><font color="#FFFF00" ><b>You are currently taking the
         following Courses</b></font></p>
       </td>
      </tr>
      <tr>
       <td width="8%" bgcolor="#C0C0C0"><font size="2">No</font></td>
       <td width="9%" bgcolor="#C0C0C0"><font size="2">Course Id</font></td>
       <td width="11%" bgcolor="#C0C0C0"><font size="2">Course Name</font></td>
       <td width="40%" bgcolor="#C0C0C0"><font size="2">Explanation </font></td>
```

```
        <td width="13%" bgcolor="#C0C0C0"><font size="2">Credits</font></td>
        <td width="9%" bgcolor="#C0C0C0"><font size="2">Professor</font></td>
      </tr>
      <jsp:getProperty name="myStudent" property="courses" />
    </table>




<!-- end of courses part-->

   <%}  if (myStudent.getOperation().equals("prof_schedule")) {%>

<!--this is the class schedule part-->

 <table border="1" width="100%" cellspacing="1" height="390" bgcolor="#00FFFF">
      <tr>
        <td width="100%" colspan="7" height="19" bgcolor="#FF0000">
          <font color="#FFFF00">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><b>CLASS
          SCHEDULE</b></td> </font>
      </tr>
      <tr>
        <td width="16%" colspan="2" height="19" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </td>
        <td width="84%" colspan="5" height="19" bgcolor="#0000FF">
          <p align="center" style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="1"
color="#FFFF00">DAYS</font></td>
      </tr>
      <tr>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">No</font></td>
        <td width="8%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Time</font></td>
        <td width="16%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Monday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Tuesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Wednesday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Thursday</font></td>
        <td width="17%" height="14" bgcolor="#0000FF">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font color="#FFFF00"
size="2">Friday</font></td>
      </tr>
      <jsp:getProperty name="myStudent" property="schedule" />
    </table>
```

```
<!--end of the class schedule part -->

  <%} if (myStudent.getOperation().equals("Personal_information")) {%>

<!-- this is the prof personal information part -->

     <table border="1" width="100%" height="542">
       <tr>
         <td width="589" height="36" colspan="2" bgcolor="#FF0000">
          <p align="center"><font color="#FFFF00"><b>Personal Information For 
          <jsp:getProperty name="myStudent" property="name" /> 
          <jsp:getProperty name="myStudent" property="lastName" />
          </b></font>
          </td>
       </tr>
       <tr>
         <td width="491" height="92">
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
           </p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">Name          
          <input readonly type="text" name="name" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="name" />">  
          M. 
          <input readonly type="text" name="midname" size="12" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="midName" />">
           
          Last Name  <input readonly type="text" name="lastname" size="12" style="background-
color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="lastName" />">
          </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Section</font><font
size="2">
                 <input readonly type="text" name="degree" size="20"
style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="section" />">
          </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
          <font size="2">
          Department
          <input readonly type="text" name="program" size="51" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="department" />">
          </font></p>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Regiment
          Unit <input readonly type="text" name="rank" size="50" style="background-color: #C0C0C0"
          value="<jsp:getProperty name="myStudent" property="regiment" />">
           </font>
          <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> 
          </td>
          <td width="98" height="92" bgcolor="#FFFFFF">
          <p align="center">
          <img          border="0"          src="/src/webadmin/<jsp:getProperty          name="myStudent"
property="userId"/>.jpg" width="70" height="70">
          </td>
       </tr>
       <tr>
```

```
            <td width="595" colspan="2" height="214"> 
          <form method="POST" action="/servlet/webadmin.Academy" onSubmit="" style="word-spacing:
0; line-height: 100%; margin: 0">
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Email       
            <input type="text" name="email" size="20" tabindex="1"
            value="<jsp:getProperty name="myStudent" property="email" />">
            </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">Dormitory:  </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Room
            No <input type="text" name="room" size="20" tabindex="2"
            value="<jsp:getProperty name="myStudent" property="room" />">
            Location <input type="text" name="LOCATION" size="20" tabindex="3"
             value="<jsp:getProperty name="myStudent" property="location" />">
             </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Home:</font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font size="2">Address 
            <input type="text" name="address" size="70" tabindex="4"
            value="<jsp:getProperty name="myStudent" property="address" />">
             </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"><font
size="2">City         
            <input type="text" name="city" size="20" tabindex="5"
            value="<jsp:getProperty name="myStudent" property="city" />">
             Province <input type="text" name="province" size="20" tabindex="6"
            value="<jsp:getProperty name="myStudent" property="province" />">
            </font></p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0">
            <font size="2"><input type="submit" value="Update" name="updateButton"></font></p>
                  <input type="hidden" name="action" value="personal_update">
            <p style="word-spacing: 0; margin-top: 0; margin-bottom: 0"> </p>
          </form>
        </td>
      </tr>
        <tr>
            <td width="595" colspan="2" height="19"> 
            <font color="#FF0000" size="2">Message :
            <jsp:getProperty name="myStudent" property="message" /></font>

        </td>
      </tr>

          <tr>
                <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">

            <td width="595" colspan="2" height="151"> 
        <font size="2"><b>PASSWORD
          CHANGE</b></font>
          <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2">Your Login
          Id is
          :           
```

```
                   
          <input readonly type="text" name="midname" size="20" style="background-color: #C0C0C0"
         value="<jsp:getProperty name="myStudent" property="login" />"> </font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Enter the
          new Password:        <input type="password"
name="firstText" size="20"></font></p>
          <p style="line-height: 100%; margin: 0"><font size="2">Confirm
          your new Password: <input type="password" name="secondText" size="20"></font></p>
                    <p style="line-height: 100%; margin: 0"> </p>
          <p style="line-height: 100%; margin: 0"><font size="2"><input type="submit" value="Change
Password" name="B1"></font></p>
                <input type="hidden" name="action" value="change_Password">
        </td>
        </form>
       </tr>
     </table>
<!------------------------------------------------>


<!-- end of personal information part-->

<!-- this is the students for prof part -->
    <%} if (myStudent.getOperation().equals("gradesForStudent")) {%>

<table border="1" width="688" height="174">
  <tr>
    <td width="678" colspan="3" height="45" bgcolor="#FF0000">
     <p align="center"><font color="#FFFF00">
     <b>The grades for <jsp:getProperty name="myStudent" property="name"/> <jsp:getProperty
name="myStudent" property="lastName" /></b></font></td>
  </tr>
  <tr>
    <td width="79" height="1" bgcolor="#C0C0C0"><b>Student Id</b></td>
    <td width="122" height="1" bgcolor="#C0C0C0"><jsp:getProperty name="myStudent"
property="userId" /></td>
    <td width="459" height="3" bgcolor="#C0C0C0" rowspan="2">
     <p align="left"><b>Which Semester grade do you want to see?</b></p>
     <form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
        <input type="hidden" name="action" value="gradesForStudent">

       <p align="left"> <select size="1" name="Semester">
        <option>All grades</option>
        <option selected>Year 2002 semester 2nd</option>
        <option >Year 2002 semester 1st</option>
        <option>Year 2001 semester 2nd</option>
        <option>Year 2001 semester 1st</option>
        <option>Year 2000 semester 2nd</option>
        <option>Year 2000 semester 1st</option>
        <option>Year 1999 semester 2nd</option>
        <option>Year 1999 semester 1st</option>
       </select>          
       <input type="submit" value="SHOW" name="B1"></p>
     </form>
    </td>
  </tr>
  <tr>
```

```
   <td width="79" height="20" bgcolor="#C0C0C0"><b>Name</b></td>
   <td width="122" height="20" bgcolor="#C0C0C0">
   <jsp:getProperty name="myStudent" property="name" /> <jsp:getProperty name="myStudent"
property="lastName" /></td>
  </tr>
  <tr>
   <td width="660" height="34" bgcolor="#C0C0C0" colspan="3">
    <p align="center"><b>The followings are &amp;quot; <jsp:getProperty name="myStudent"
property="semester" /> &amp;quot;
     grades&amp;nbsp;</b></p>
   </td>
  </tr>
</table>
<table border="1" width="689" height="77">
        <tr>
   <td width="92" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">

    <font color="#FFFF00"><b>Professor</b></font>
 </td>
   <td width="124" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Course Id</b></font>
 </td>
   <td width="107" height="1" rowspan="2" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Course Name</b></font>
 </td>
   <td width="424" height="13" colspan="4" bgcolor="#0000FF">
   <p align="center"><font color="#FFFF00"><b>----------Grades----------
</b></font></p>
   </td>
        </tr>
        <tr>
   <td width="111" height="1" bgcolor="#0000FF" valign="middle">
      <font color="#FFFF00"><b>First Exam</b></font>
        </td>
   <td width="134" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Second Exam</b></font>
        </td>
   <td width="65" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Final</b></font>
        </td>
   <td width="99" height="1" bgcolor="#0000FF" valign="middle">
    <font color="#FFFF00"><b>Course Grade</b></font>
        </td>
       </tr>



   <jsp:getProperty name="myStudent" property="courses" />
    </table>

<!-- end of students for prof -->



    <%} if (myStudent.getOperation().equals("companyInfo")) {%>
```

```
  <table border="1" width="689" height="161">
 <tr>
  <td width="679" colspan="5" bgcolor="#FF0000" height="40">
   <p align="center"><font color="#FFFF00"><b>The
   Company Information for
   <jsp:getProperty name="myStudent" property="name" />
   <jsp:getProperty name="myStudent" property="lastName" />
    </b></font>
  </td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Company name</b></td>
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="companyName" />
  </td>
  <td width="225" height="19" colspan="2"></td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Regiment Commander</b></td>
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="regComName" /></td>
  <td width="225" height="19" colspan="2"></td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Battalion Commander</b></td>
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="batComName" />
  </td>
  <td width="225" height="19" colspan="2"></td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Company Commander</b></td>
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="compComName" />
  </td>
  <td width="225" height="19" colspan="2"></td>
 </tr>
 <tr>
  <td width="655" height="5" colspan="5" bgcolor="#FFFF00">
   <p align="center"> </p>
  </td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Student Id</b></td>
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="userId" />
  </td>
  <td width="126" height="76" bgcolor="#C8C8C8" rowspan="2"><b><font
color="#000000">Remaining
   Disciplinary points</font></b></td>
  <td width="99" height="76" bgcolor="#C8C8C8" rowspan="2"><font color="#000000">
  <jsp:getProperty name="myStudent" property="points" />
  </font></td>
 </tr>
 <tr>
  <td width="182" height="19" colspan="2"><b>Student Name</b></td>
```

```
  <td width="248" height="19">
  <jsp:getProperty name="myStudent" property="name" />
  </td>
 </tr>
 <tr>
  <td width="679" colspan="5" bgcolor="#0000FF" height="40">
   <p align="center"><font color="#FFFF00"><b>AWARDS</b></font></td>
 </tr>
 <tr>
  <td width="84" height="19"><b>Award No</b></td>
  <td width="98" height="19"><b>Award Type</b></td>
  <td width="248" height="19"><b>Result</b></td>
  <td width="231" height="19" colspan="2"><b>Award Date</b></td>
 </tr>
 <jsp:getProperty name="myStudent" property="awards" />
  <tr>
  <td width="679" colspan="5" bgcolor="#0000FF" height="40">
   <p align="center"><font color="#FFFF00"><b>PUNISHMENTS</b></font></td>
 </tr>
 <tr>
  <td width="84" height="19"><b>Punishment No</b></td>
  <td width="98" height="19"><b>Punishment Type</b></td>
  <td width="248" height="19"><b>Result</b></td>
  <td width="126" height="19"><b>Points Taken</b></td>
  <td width="99" height="19"><b>Punishment Date</b></td>
 </tr>
 <jsp:getProperty name="myStudent" property="punishments" />
 </table>
  <%} %>
   <p> 
   <p> 
   <p> 
   <p> 
   <p> 
   <p> </td>
 </tr>
</table>



</body>

</html>
```

10. Source Code of the Login JSP File

```
<!--File name: Login.jsp
   Purposes : To handle the users logins.-->

<%@ page errorPage="ErrorPage.jsp" %>
<HTML>
<HEAD>
<jsp:useBean id="myLogin" scope="session" class="webadmin.Communicator" />
<jsp:setProperty name="myLogin" property="*" />
```

```
<TITLE>
Login
</TITLE>
</HEAD>
<body leftmargin="6">
<table border="0" width="100%">
  <tr>
    <td width="100%">
      <p       align="center"><img       border="0"       src="/src/webadmin/academy.gif"       width="770"
height="100"></td>
  </tr>
  <tr>
    <td width="100%"></td>
  </tr>


</table>
<table border="0" width="100%">
  <tr>
    <td width="24%">

      <p> </td>
    <td width="76%">
<form method="POST" action="/servlet/webadmin.Academy" onSubmit="">
  <p> <jsp:getProperty name="myLogin" property="message" />
  <p> Enter your login Id and
  password</p>
  <p>login Id     <input name="loginId" size="20"></p>
  <p>password <input type="password" name="password" size="20"></p>
  <input type="hidden" name="action" value="LOGIN">

  <p><input type="IMAGE" src= "/src/webadmin/login.gif" value="LOGIN" name="action">
  </p>
</form>
      <p> </td>
  </tr>
  <tr>
    <td width="24%">
    </td>
    <td width="76%"></td>
  </tr>
</table>
<p> </p>
</body>

<HTML>
```

11. Source Code of the LogOff JSP File

```
<!--File name: LogOff.jsp
   Purposes : To handle the users log offs.-->

<%@ page errorPage="ErrorPage.jsp" %>
<HTML>
<HEAD>
<jsp:useBean id="myLogOff" scope="session" class="webadmin.Communicator" />
```

```
<jsp:setProperty name="myLogOff" property="*" />

<TITLE>
Login
</TITLE>
</HEAD>
<body leftmargin="6">
<table border="0" width="100%">
  <tr>
    <td width="100%">
      <p       align="center"><img       border="0"       src="/src/webadmin/academy.gif"       width="770"
height="100"></td>
  </tr>
  <tr>
    <td width="100%"></td>
  </tr>


</table>
<table border="0" width="100%">
  <tr>
    <td width="24%">

      <p> </td>
    <td width="76%">

  <p> <jsp:getProperty name="myLogOff" property="name" /> </p>
  <p>You successfully logged off.</p>
  <p>The time is   <jsp:getProperty name="myLogOff" property="time" /></p>


      <p> </td>
  </tr>
  <tr>
    <td width="24%">
    </td>
    <td width="76%"></td>
  </tr>
</table>
<p> </p>
</body>

<HTML>
```

12. Properties File

```
drivers=sun.jdbc.odbc.JdbcOdbcDriver
logfile=D:/java/tez/log.txt

myThesis.url=jdbc:odbc:oracle
myThesis.user=SCOTT
myThesis.password=TIGER
myThesis.initconns=3
myThesis.maxconns=10
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ayers, D., Bergsten, H. and et al., *Professional Java Server Programming*, Wrox Press Ltd., 2000.

Akbay, M., Lewis, S., *Design and Implementation of an Enterprise Information System Utilizing a Component Based Three-tier Client/Server Database System,* NPS, 1999.

Beckhard, R., *A Model for Executive Management of Transformational Change,* Human Resource Strategies, London:Sage/Open University, 1992

CERT (Coordination Center Software Engineering Institute Carnegie Mellon University Pittsburgh), *Results of the Security in ActiveX Workshop Pittsburgh,* 2000, Online, Internet. Available: http://www.cert.org/reports/activeX_report.pdf

Comer, D., *Computer Networks and Internets*, Prentice Hall, 2001

Davidson, L., *Professional SQL Server 2000 Database Design,* Wrox, 2001.

Edwards, J. *Three-tier Client/Server At Work*, revised edition, John Wiley & Sons, 1999.

Elmasri, R., Navathe, S., *Fundamentals of Database Systems,* Second edition, Addison Wesley, 1994.

Gallaugher J. M., Ramanathan S. C., *Choosing a Client/Server Architecture, A Comparison of Two- and Three-Tier Systems*, Information Systems Management, Vol. 13, page 7-13, 1996.

Goodyear, M., *Enterprise System Architectures*, CRC press, 2000.

Gross, C., *Taking the Splash Diving into ISAPI Programming*, Microsoft, 1997, online, Internet. Available: http://www.microsoft.com/mind/0197/isapi.htm

Haga, W., *Personal Conversation,* 2002.

Hall, M., *Core Servlets and Server Pages,* Prentice Hall, 2001

Mabey, C. and Iles, P., *Strategic Integration of Assessment and Development Practices:Succession Planning and New Manager Development,* Human Resource Management Journal, 1993

NCSA, The National Center for Supercomputing Applications , *CGI Common Gateway Interface,* The National Center for Supercomputing Applications (NCSA) at the University of Illinois, 1999, online, Internet. Available: http://hoohoo.ncsa.uiuc.edu/cgi/

Netcraft, *Web Server Survey*, September 2001, online, Internet. Available: http://www.netcraft.com/survey

Orfali, R. H., Edwards, J., *The Essential Client/Server Survival Guide*, John Wiley & Sons, 1996.

Pressman, R. S., *Software Engineering a Practitioner's Approach*, McGraw Hill, fifth edition, 2001

Ramakrishnan, R., Gehrke, J., *Database Management Systems,* McGraw Hill, second edition, 2000.

Reese, G., *Database Programming with JDBC and JAVA,* O'Reilly, 2000

Sun Inc.,Comparing Methods for Server-Side Dynamic Content (White Paper), Sun.com, 2000, online, Internet. Available: http://java.sun.com/products/jsp/jspservlet.html

Taylor, A., *JDBC Developer's Source,* Second edition, Prentice Hall, 1999.

White, S., Hapner, M., *JDBC 2.1 API,* Sun, 1999, online, Internet. Available: http://java.sun.com/j2se/1.3/docs/guide/jdbc/spec2/jdbc2.1.frame.html

White, S., Fisher, M, and et al, *JDBC API Tutorial and Reference,* Addison Wesley, second edition, 1999.

Whitten, J.L., Bently, L. D., Dittman, K. C., *System Analysis and Design Methods,* McGraw-Hill, fifth edition, 2001.

INITIAL DISTRIBUTION LIST

1. DEFENSE TECHNICAL INFORMATION CENTER
   8725 John J.Kingman Road, Ste 0944
   Ft. Belvoir, VA

2. DUDLEY KNOX LIBRARY
   Naval Postgraduate School
   Monterey, CA

3. DENIZ KUVVETLERI KOMUTANLIGI
   Personel Daire Baskanligi
   Bakanliklar 06410
   Ankara, TURKEY

4. DENIZ KUVVETLERI KOMUTANLIGI
   Kutuphanesi
   Bakanliklar 06410
   Ankara, TURKEY

5. DENIZ HARP OKULU
   Kutuphanesi
   Tuzla 81740
   Istanbul, TURKEY

6. KARA HARP OKULU
   Kutuphanesi
   Bakanliklar 06410
   Ankara, TURKEY

7. HAVA HARP OKULU
   Kutuphanesi
   Yesilyurt
   Istanbul, TURKEY

8. CHAIRMAN
   Code CS
   Naval Postgraduate School
   Monterey, CA

9. CHAIRMAN
Code C4I
Naval Postgraduate School
Monterey, CA

10. C. THOMAS OTANI
Code CS/CSTO
Naval Postgraduate School
Monterey, CA

11. WILLIAM HAGA
Code GSBPP/GBHG
Naval Postgraduate School
Monterey, CA

12. LTJG. RASIM TOPUZ
Envanter Kontrol Merkez Komutanligi
Golcuk 41650
Kocaeli, TURKEY